1972-06

Interactive logic laboratory

Johnson, Robert Lee, Jr.

INTERACTIVE LOGIC LABORATORY

Robert Lee Johnson

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INTERACTIVE LOGIC LABORATORY

by

Robert Lee Johnson, Jr.

Thesis Advisor: George A. Rahe

June 1972

INTERACTIVE LOGIC LABORATORY

by

Robert Lee Johnson, Jr.
Lieutenant, United States Navy
B.S., Naval Postgraduate School, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

ABSTRACT

This thesis documents an investigation into the use of a computer graphics terminal to demonstrate the basic concepts of logical design. The areas of computer-assisted instruction, computer graphics, and computer-aided design are reviewed prior to the discussion of the creation of the INTERACTIVE LOGIC LABORATORY. The program is implemented on the Adage Graphics Terminal - 10 (AGT-10) of the Naval Postgraduate School Computer Laboratory.

The main emphasis of the program discussion is on the degree of interaction achieved by the program and its possible use as a learning aid for students of basic logical design courses. A bipartite graph is used to depict the network topology of the logic circuit and the program is quite successful in the simulation of simple logic circuits.

TABLE OF CONTENTS

I. INTRODUCTION

The value of laboratory work has long been recognized in the physical sciences. This most important adjunct to the learning process has both the demonstrative power of presenting classroom concepts in a real-world setting as well as allowing the student to learn by doing. The experience gained by actual experimentation in the laboratory serves the dual purpose of aiding the learning process and preparing the student for more rigorous research in later work.

If Alfred North Whitehead's triad of learning -- a stage of romance, a state of precision, a stage of generalization [ref. 1] - is accepted, the value of the laboratory becomes particularly evident. Thought-provoking demonstrations, quite easily presented in the laboratory environment, aid immensely in arousing student interest in a topic. The ability for precise analysis can be developed by student experimentation. Thus leading to a lesser reliance on previously learned specifics, a better understanding of the fundamental concept and the ability to extrapolate to its implication.

The value of laboratory experience is also well appreciated in the computer sciences as evidenced by the huge investment institutions of higher learning have made in computer facilities, both for actual research and student experimentation. For instance, a dual-processor IBM Model 360-67 is installed at the Naval Postgraduate School and this facility is utilized for both administrative work as well as extensive student and faculty research. Another computer facility implemented at the school is a hybrid computer comprised of an XDS-9300 processor

interfaced with a Comcor CI-5000 analog computer and two Adage (AGT-10) Graphics terminals.

It is impossible for a student at the Naval Postgraduate School to complete any of the various courses of study without some degree of exposure to one of these computation facilities. Yet, none of these computer installations is currently adapted for demonstration and experimentation in one of the most basic areas of computer science - logical design. Various logic demonstrators have been marketed, in fact the Comcor CI-5000 has the capability of being used as a logic demonstrator. However, the use of this analog computer for demonstration of basic logic design concepts has not been explored due to the high cost associated with this implementation.

The lack of adequate laboratory facilities in a basic logical design course has been apparent to several of the members of the faculty at the Naval Postgraduate School. Various approaches to the problem have been tried. One professor assigned work on the CDC-160 which involved using that machine's bit manipulative capabilities, both logical and arithmetic, to simulate the various logical operations discussed in class. A more extensive effort to implement a meaningful laboratory environment for basic logical design consisted of a computer program written for the IBM-360 which performed detailed digital machine simulation at the bit-handling level through control unit level. This program [ref. 2] was designed to provide the student with an operating model of the digital computer capable of both demonstrating computer operations and allowing realistic experiments in logical design.

While the program was a significant improvement over any previous laboratory capability, it still was lacking in one important respect -- the simulation of a <u>hardware</u> - oriented subject was done using software.

The published article recognized "The need for an early course introducing the digital computer as a hardware system." [2] Yet student programming ability in a higher level language was presupposed as a requirement for use of the digital machine simulation program.

With this background it was desired to use existing facilities at the Naval Postgraduate School to investigate the implementation of a meaningful instructional vehicle for presenting the basic concepts of logical design which did not require student programming ability in a higher-level language. The criteria by which the effectiveness of the implementation would be measured were:

1) The ability of the program to clearly demonstrate, in familiar terms, the basic concepts of logical design.
2) The ability of the program to allow for student experimentation.
3) Ability of the program to be utilized by a large number of students with minimal instruction and minimal interference with other computer users.

Thus the problem consisted of investigating the various requirements of such a program, determining a suitable set of equipment for implementation, and finally creating such a program to demonstrate the feasibility of the concept. The intent of this thesis was not to actually implement the program at the classroom level, but to investigate the construction of such a program in sufficient depth to allow subsequent extension of the concepts considered in the thesis to a general classroom implementation.

II. BACKGROUND

A. EDUCATIONAL AIDS

Many different educational aids have been developed and marketed over the past few years. The teaching aids committee of The American Society for Engineering Education (ASEE) was established to review and recommend various educational aids to institutions of higher learning. In doing so, they established definitions of various types of educational aids which were considered pertinent to this study. In particular, they subsumed the general term "educational aid" into the more comprehensive categories of teaching aids, learning aids and training aids.

Of special importance to this project was the ASEE definition of a learning aid as "a device that helps in the understanding of a fundamental of engineering or science and which creates within the student a desire to pose and solve a problem by an application of the fundamental involved."

Of the three categories defined, the committee stressed most heavily the importance of learning aids in the educational process, stating in part:

"The old adage, 'Experience is the best teacher' might be rephrased thus, 'Experience is the best way to learn.' All of the experimental evidence from educational psychology tends to substantiate this concept... student participation, either purely mental or a combination of physical and mental, is the key to successful efficient learning."

Hence in the construction of the proposed educational aid, it was mandatory that it permit direct student experience in the actual design

of computer logic circuits. This would most profitably be done within a framework that would enable the student to construct a logic circuit in familiar form using standard symbology. Further experience would be obtained by allowing student control of the inputs to the constructed circuit and observation of the responses of the circuit to these inputs. The major emphasis being on student control of variables and analysis of results.

B. SUMMARY OF PREVIOUS RESEARCH

The above requirements posed by an investigation of the basic considerations in the design of an effective educational aid when considered together with the requirement for clarity and ease of use indicated a graphical approach to the problem. The expression, "One picture is worth a thousand words" has been used (almost too) extensively to characterize the value of graphical display, but nonetheless underscores the impact of this means of presenting information. Additionally, the requirements for student interaction with and control of the learning aid strongly suggested that it could best be constructed by using a digital computer with graphical input and output capability. The computer would be programmed to accept and display student inputs, perform analysis on these inputs, and display the results of the student's circuit design efforts.

Hence the fields of computer graphics in general and computer-aided design in particular were investigated as a prelude to construction of the program. The term, "computer graphics," has been defined [ref. 3] as that set of computer techniques and applications wherein data is either presented or accepted by a computer in graphical form. "Computer-aided design" (CAD) as the name implies, specifies the use of a

computer as an assistant in the design of some entity. While computer graphics is a general enough term to encompass all computer systems using graphical display, computer-aided design usually implies a significant degree of man-computer interaction in the symbiotic relationship perceived by Licklider [ref. 4]. The designer specifies his ideas to the computer in graphical form, uses the computer's significant computational ability to perform some analysis of the design, possibly modifying the design for re-analysis. While the purpose of the proposed program did not include its specific use as a design aid for actually fabricating logic circuits, it was felt that the computer-aided design approach would be most successful in exposing the student to design concepts as applied to logic circuits.

1. Computer-Aided Design (CAD)

Early research in the field of computer-aided design centered at the Massachusetts Institute of Technology. Professor Steven A. Coons of the mechanical engineering department at that institution published a paper in the early 1960's outlining the requirements for a computer-aided design (CAD) system [ref. 5]. The paper traced CAD development from its genesis in the Automatic Programmed Tool System to the level of sophistication displayed by Sutherland's SKETCHPAD [ref. 6].

In examination of the design process itself, Coons saw "a few engineers performing highly creative tasks at the beginning, coupled with a very large number of draftsmen and technicians, who perform relatively uncreative tasks over a fairly long period of time." [5] He further envisioned that this process could be vastly improved by using a computer with a graphical capability to accept, interpret, and remember shape descriptive information. Additionally, the computer system

must have the ability to perform the mathematical analysis necessary to evaluate the design with respect to predetermined objectives.

In support of his vision of improvement in the design process by computer, Coons enumerated several CAD system benefits:

1) Emphasis on interaction and inter-communication between design users.
2) Dynamic display of time-varying systems.
3) Use of more accurate mathematical models, allowed by increased computational power.
4) Exponential design rate, with subelements of design saved by computer.
5) Use of the same basic structure by different design disciplines.

No discussion of computer graphics or computer-aided design would be complete without reference to the pioneering work of I. E. Sutherland, also at MIT. The SKETCHPAD system [6] was the first to demonstrate the effective use of an interactive display console to accept inputs and display outputs in graphical form and control the sequence of program execution. Sutherland's program was built around a powerful data structure which allowed for representation of display elements, labeling of various parts of the display with alphanumerics, and representation of display topology. Analysis was accomplished in the program by the use of mathematical conditions (called "constraints,") on parts of the drawing. The addition of design constraints as well as geometrical constraints gave SKETCHPAD a significant design capability, although at the time of publication, Sutherland's program had not demonstrated the ability to design electrical circuits.

2. Circuit Design Programs

The view of the nature of the problem as a logic circuit design task indicated a review of extant computer programs constructed for this purpose. Some examples of user-oriented circuit analysis programs reviewed were Electronic Circuit Analysis Program (ECAP) [ref. 7] and Continuous Systems Modeling Program (CSMP) [ref. 8] both IBM circuit analysis applications. Also investigated were Automated Engineering Design of Networks (AEDNET) [ref. 9] and Circuit Analysis (CIRCAL) [ref. 10] by J. Katzenelson and M. L. Dertouzos respectively of MIT. Of most pertinence to the construction of the logic demonstrator was Dertouzos's paper, "Introduction to On-Line Circuit Design." [ref. 11].

In this paper, Dertouzos characterized on-line circuit design as a design dialog with short interaction delays. He further listed various requirements which must be met if the implementation of a circuit design program is to be truly interactive. These include:

1) An editing requirement to accomplish inputting of information such as network description, element description, variable values, etc.
2) An output requirement to convert computer generated information into a form suitable for transmission to the user.
3) A definitional requirement to enable users to build subelements of a circuit design to be used in later more complex circuits.
4) An informational requirement to provide the system with necessary control information to execute the program.
5) A diagnostic requirement to enable the user to discover mistakes in his use of the program or design.

In reference 11, Dertouzos also discussed in detail the internal structure of on-line programs for circuit design emphasizing the need for storing of information and insuring proper information flow between various program segments. The importance of a comprehensive data structure capable of representing the topology of the network was presented along with the benefits of such a data structure. These include:

1) Efficient use of storage.
2) Efficient application of algorithms.
3) Use of operators which are independent of the size and structure of stored information.
4) Efficient representation and processing of recursive constructs.

The analysis portion of a typical on-line circuit design program was considered in light of the requirements for interaction, such as premature termination of the analysis by the designer or changes in the course of the analysis designated by the designer. The nature of the on-line approach to circuit design was shown to be an adaptive type of process as opposed to one with predetermined structure.

3. Circal: An On-Line Circuit Design Program

The above requirements for a circuit design program were met in the CIRCAL program implemented at MIT by Dertouzos [10]. Several facets of the program applied directly to the development of the proposed logical design program. The program itself had three distinct versions (CIRCAL-0, CIRCAL-1, CIRCAL-2), each capable of handling an increasingly complex electrical network. The program operated on-line on a modified IBM 7094 under the Project MAC system using either graphical or teletype modes. Of special interest were the use of a grid mesh, superimposed on

the display, with the restriction that circuit elements could lie only on the grid intersections. Connection of individual circuit elements was done using analysis of typewritten commands.

In consideration of the interactional requirements listed above, the program was structured into three main segments. These were:

1) A DEFINE mode for circuit elements and waveforms.
2) An INPUT/EDIT mode for forming or changing network connections and specifying element values.
3) A TEST/OUTPUT mode for observing response of the designed circuit to the specified inputs.

Consideration was given to the importance of the data structure and analysis methods in the overall effectiveness of the system. Also of importance was the observation that "the 'input/edit' and 'define' functions of any on-line circuit design system are in principle independent of the methods used for network analysis." [10].

4. Computer-Assisted Instruction (CAI)

In view of the didactic nature of the proposed program, an investigation of the field of computer-assisted instruction was deemed appropriate. Computer-assisted instruction evolved from the concept of programmed instruction first articulated by S. L. Pressey at Ohio State University in the 1920's. This learner-centered method of instruction presents new information to the student in the form of incremental steps with constant review and testing to reinforce learning. The method did not earn general acceptance until the need for reinforcement in learning was underscored by the research of B. F. Skinner at Harvard University in the middle 1950's. [ref. 12].

G. M. and L. C. Silvern [ref. 13] listed the criteria governing programmed instruction which are now generally accepted as:

1) Instruction is provided without the presence of a human instructor.
2) The learner progresses at his own rate (conventional group instruction, films, television and other fixed-format media do not satisfy this criterion.)
3) Instruction is presented in small incremental steps requiring frequent response by the learner.
4) There is a participative, overt interaction or two-way communication between learner and instructional program.
5) Learner receives immediate feedback informing him of his progress.
6) Reinforcement is used to strengthen learning.

Although the methods of programmed instruction were unusually well suited for computer implementation, their appearance before the general availability of computers to educational institutions lead to initial textual implementation. The original structure of a programmed instructional text was essentially linear in nature. The student was presented an increment of information then tested on the concept involved. If the student responded incorrectly, he was given a simplified and expanded version of the same information and allowed to proceed. More advanced programmed instruction methods soon developed with a branching structure capable of allowing brighter students to progress at a faster rate and tailoring the remedial information to the mistaken response given.

Computer-assisted instruction then implies the implementation of programmed instructional concepts on a digital computer. The interaction, feedback, and reinforcement specified above make an interactive graphical approach especially well suited to computer-assisted instruction.

5. Basic Logical Design

By far, the oldest area of interest to the proposed project was that of logical design. In 1854 George Boole, an English mathematician, published his classic book: An Investigation of the Laws of Thought on Which Are Founded the Mathematical Theory of Logic and Probabilities. Proceeding from his basic investigation of classical logic, Boole derived a "logical algebra" which today bears his name.

The ability of boolean algebra to adequately describe the behavior of relay switching circuits was first recognized by C. W. Shannon, also of MIT. In his Masters Thesis: "A Symbolic Analysis of Relay and Switching Circuits," [ref. 14] Shannon showed that any circuit consisting of combinations of switches and relays could be represented by a set of mathematical expressions. He further showed that these expressions were exactly equivalent to the algebra derived by Boole in the field of symbolic logic. Thus boolean algebra finds much application in the design of digital computer systems composed of storage elements and their associated circuitry for switching from one state to another.

Boolean algebra differs from ordinary algebra in some fundamental ways. As in ordinary algebra, letters are used as terms in boolean expressions but their meaning is different. Boolean variables can take on only two distinct values (usually represented by the binary numbers 0 and 1). Thus boolean variables are useful for depicting the existence

or non-existence of a given condition, such as a switch being open or closed or a statement being true or false. Boolean functions may be formed by using a number of different operators. However, all of these operators are derivable from sets of primitive boolean operators. One commonly known set consists of intersection (AND), union (OR), and negation (NOT).

At the basic level taught in an introductory course, logical design involves the use of boolean primitives to describe information flow in a digital computer. Additionally, since a digital computer is a finite state machine, information storage requirements must be considered. Temporary storage (registers) or permanent storage (core) is accomplished in a computer by some type of bi-stable device. For design applications, register storage is usually of most interest and is accomplished by means of a flip-flop (bi-stable multivibrator).

Implicit in the assumption of a binary storage element is the ability for this element to be able to change state. Hence the value of Shannon's thesis is the ability to describe the conditions necessary for the switching of element states in boolean terms. The types of flip-flops thought to be of most application in a basic logical design course were the clear-set flip-flop, clear-set-trigger flip-flop and J-K flip-flop. State diagrams for these various flip-flops as well as truth-table representations of the boolean primitives AND and OR are found in the Appendix.

III. PROGRAM IMPLEMENTATION

A. COMPUTER SELECTION

With the above background information reviewed and under the assumption that an interactive learning aid for the design of basic computer circuits would be of value in the instruction of Naval Postgraduate School students, the actual construction of the program began. Of primary concern was the selection of the computer installation upon which to implement the program. Prior statements emphasizing the need for interaction and graphical display inherent in the problem narrowed the choice to the Xerox Data Systems 9300/Adage Graphics Terminal - 10 (AGT-10) system in the Naval Postgraduate School Computer Laboratory. It was felt that in achieving "minimal interference with other computer users", the logic design program would be best implemented on the AGT-10 system using this system's "stand alone" capability.

The main consideration in the selection of this computer facility was the significant ability of the Adage Graphics terminal to be programmed for an interactive instructional application. The graphical display capability of this equipment, coupled with the interfaced user communication devices provided an ideal research vehicle for this computer graphics task. However, this is not proposed as a cost-effective way to build a learning aid.

A large, fast processor (XDS 9300) capable of accepting higher level language programs, tied to a separate smaller computer (AGT-10) responsible for the graphical display was a concept used in many of the circuit design implementations cited above. However, in view of the

program's proposed use as a learning aid, it was felt the increased availability of the program to potential users offered by "stand alone" implementation would be of value in earning student acceptance of the program.

The "stand alone" capability of the AGT-10 is available by virtue of the fact that the Adage graphics unit has its own processor. The graphics facility provided by the AGT-10 system is comprised of a high speed, high precision vector generator with cathode ray tube (CRT) display [ref. 15]. The CRT has a 12 by 12 inch display area, with a smaller area of high resolution. A stroke-type character generator is also available for CRT display of alphanumerics. The graphics console of the AGT system has additional devices incorporated which facilitate graphical communication with the user. Devices of special interest to this program were the function switch box, light pen and alphanumeric keyboard.

The DPR-2 Digital PRocessor associated with the Adage Graphics Terminal system is a general purpose digital computer with extensive transfer logic and addressing capabilities. The processor has a 30 bit word length, memory cycle time of two microseconds and 8 K memory size. Additional random-access memory is provided by the DMS-2 Disk Memory Subsystem.

The main software support for the AGT-10 system is the Adage Extendable Program Translator (ADEPT). ADEPT is an open-ended string substitution macro translator capable of producing relocatable machine language code. Two passes of the source language (ATEXT) are made to allow unlimited forward references to symbolic addresses. Other features of the ADEPT translator are:

1) Automatic definition of location symbols.

2) Parameter assignment statements.

3) Macro nesting capability.

4) Conditional translator capability.

5) Definite and indefinite repeat statements.

A unique feature of ADEPT allows definition of "action operators" in addition to those already present in the ADEPT translator; thus allowing the programmer to extend the language to his own needs.

B. FLOW - OF - CONTROL

Preliminary to the actual coding of the program in the ADEPT language, a consideration of overall program flow of control and data structure was undertaken. It was decided to divide the program into three main modes (as in CIRCAL). These were DEFINE, CONNECT, and ANALYZE. Student specification of the circuit, selection of analysis, and observation of circuit response to specified input values would be accomplished within this main framework. In addition, a brief instructional mode would be provided within which the student would be shown necessary information to operate the program. A more expanded instructional mode would display various basic logic circuits with inputs specified and outputs displayed with instructive comments. Program flow of control would allow the student to specify various inputs to a particular circuit and observe its response, change the circuit structure by addition or deletion of circuit elements, or change the interconnection of these elements. The instructional modes would be capable of selection from any point in the program and were intended to be comprehensive enough to allow the student to operate the program without any additional instruction.

C. DATA STRUCTURE

Of upmost concern at the outset was the selection of a data structure which was sufficiently powerful to represent the display structure and topological structure of a logic circuit, yet was capable of straightforward implementation at the relatively low programming level of the ADEPT language. In most of the circuit analysis programs reviewed, the data structure used was some type of ring structure, involving a complex system of pointers. The complete ring structure requires having pointers from each data item to the preceeding and following item. Unlike a list, the structure is closed by also having pointers between the first and last data items. The inherent advantages of this type of structure are ease in searching for data items and the ability to represent multidimensional concepts wherein one data item is a member of more than one ring. Connectivity relations, set membership, and terminal node identifications are especially well represented in this manner. This concept was implemented by Sutherland in his "generic" data structure which grouped elements of the drawing by common properties [6].

While the ring data structure has these circuits representation benefits, it also creates implementation problems in an assembly level language such as ADEPT. In order to handle the more sophisticated structure, many primitive operations on the structure itself must be programmed at the machine level whereas this is not a requirement if a higher level language is used [ref. 16].

Much recent work in the area of computer data structures has preceeded from the basic foundations of graph theory. Robin Williams [ref. 17] has shown the advantages of the graph theory approach to preserve the relationships and logical associations that exist among data

items in a computer program. One such approach involves the use of a bipartite graph. The bipartite graph consisting of two distinct types of vertices (in this case inputs and outputs) is especially well suited for representation of logic circuits. This graph has been defined [ref. 18] as one whose vertices can be partitioned into two disjoint sets in such a way that every edge has its first end point in one set of vertices and the other end point in the remaining vertex set.

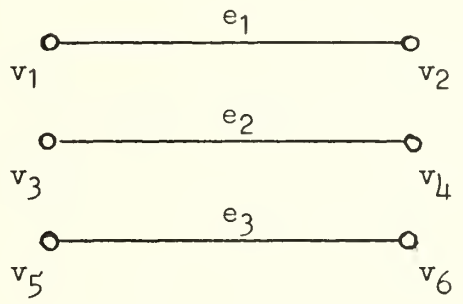

FIGURE 1

A Bipartite Graph

The circuit representation advantage offered by this approach to the data structure is that each gate may be considered as the intersection of two distinct types of pointers. Thus, the circuit is completely represented at each element by a set of pointers for inputs and one for outputs.

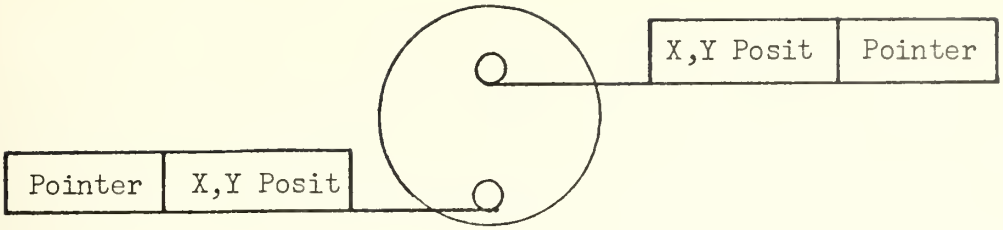

FIGURE 2

Graph Theory and Data Structure
Representation of an Arbitary Circuit Element

Another benefit of this method of representing circuits is that it does not pose a limit on the number of inputs or outputs which may be connected to any specific gate.

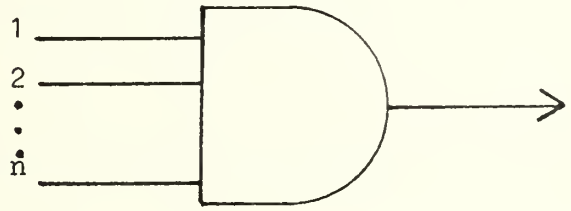

FIGURE 3

AND Gate with n Inputs

It was decided to use the bipartite graph method of circuit representation in the computer program. However, the problem of depicting this structure at the ADEPT level remained. Thus an older means of representing data structure was implemented involving establishment of table structures in contiguous areas of memory. Since the number of circuit nodes the program was designed to handle was limited, these tables could be made a fixed length. Pointer structure was implemented

at the table level in that the head item in a data table pointed to the last item and vice versa, but internal pointers within each table were not present. These were represented by the contiguity of the table. The pointers representing the edges of the bipartite graph were placed in these tables as were the X,Y coordinates representing circuit element location.

This model of the topology of the circuit consisting of direct access data sets in fixed length blocks has several benefits. It is extremely economic in its use of storage, data access time is short and the capacity of the tables can be easily increased. An additional benefit brought to light by this implementation is that display tables can be "preloaded" with display control instructions. This is especially important in the display of character information for establishing size, brightness, and italics control information.

D. MAIN PROGRAM ROUTINES

The INTERACTIVE LOGIC LABORATORY is composed of five distinct ADEPT programs designed to accomplish the objectives discussed above. The separate relocatable versions of these programs are linked together at execution time along with system routines FIN (for checking function switches) and AMRMX (for teletype interface). The five programs are hierarchically related as follows: The main program called LIL is responsible for displaying all of the user-entered and program-generated information. LIL calls four subordinate routines: LOGMM, CONCT, ANALR and INTRO based on the output of the user's light pen. LIL enables this instrument for "hits" on the text words DEFINE, CONNECT, ANALYZE and INTRODUCTION and branches to the appropriate subprogram. Thus, LIL is also responsible for the overall flow of control as specified by the user.

1. Define Mode

The ADEPT program LOGMM accomplishes the circuit element definition objective. LOGMM creates the display of the logic circuit by allowing the user to draw AND gates, OR gates, inverters and flip-flops on a 5 by 5 grid displayed on the CRT. (Only the intersections of the grid which represent possible circuit node locations are drawn as dots). A "menu" of available circuit elements is displayed at the bottom of the presentation and each menu item is labeled with the appropriate function switch which will cause a copy of that element to be drawn.

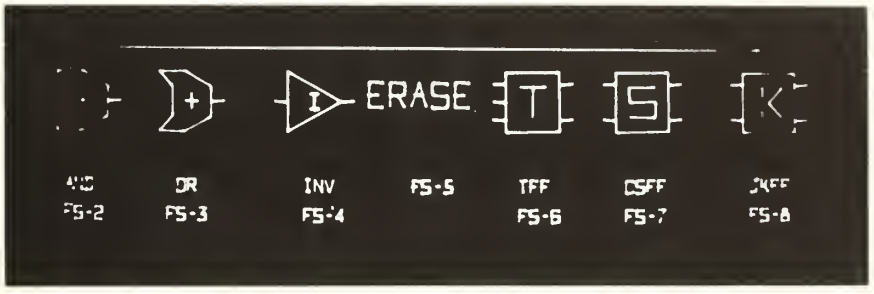

FIGURE 4

"Menu" Display

Actual circuit construction is accomplished by the user with the light pen. This instrument is used to select the dot where the circuit element is desired. Selection is indicated by the appearance of a square cursor around the dot. Depressing one of the appropriately labeled function switches chosen from the menu causes an instance of that circuit element to be entered into the programs display table called TBLO.

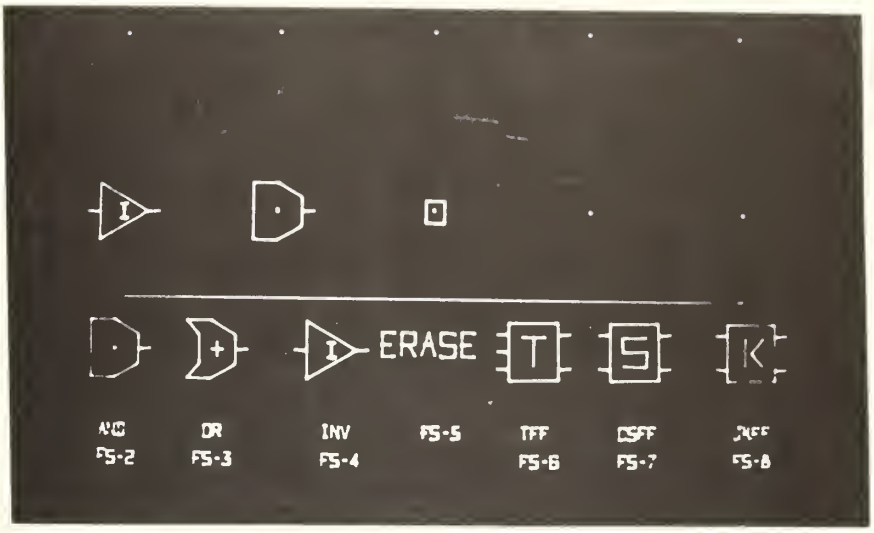

FIGURE 5

Entering Circuit Elements in DEFINE Mode

X and Y display coordinates are entered into the display table along with types of gate in the following format:

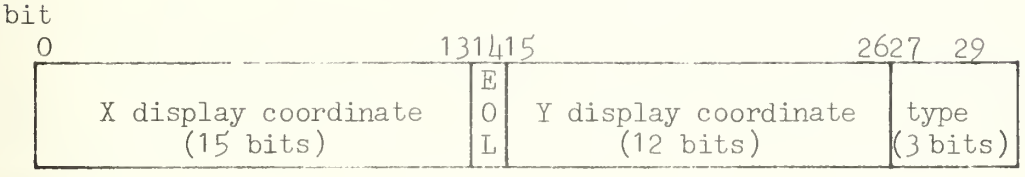

FIGURE 6

TBLO Table Entry

The top entry in the TBLO Table is a pointer to the last entry in the table. The end-of-list bit (abbreviated EOL above) is not used. The gate types are entered according to the following table:

000--not used
001--AND gate
010--OR gate
011--inverter
100--clear-set flip-flop
101--toggle flip-flop
110--J-K flip flop
111--not used

TABLE 1

Gate Types Entered in TBLO

Removal of a circuit element is accomplished by selecting an existing element with the light pen (a similar cursor indication notifies user of selection). This is accomplished by removing the erased entry from the TBLO table and moving all subsequent TBLO entries up one location. The pointer to the last entry is adjusted to point to the new last entry in the table. Upon exit the program returns to the main display mode.

2. Connect Mode

The program CONCT establishes the topological structure of the circuit. Its major function is to build the display table for the connecting lines between circuit elements previously entered in the DEFINE mode. Function switches are implemented in this mode for moving the cursor and for drawing lines to represent the circuit connections. The created lines are entered into a display data set called DATA1 and refreshed continually. Additional function switches allow the user to select any point on the screen to which he has previously moved the cursor. The points may be selected in the ordered entered (select forward) or in reverse order (select backward). This feature aids the user in "hooking" a line for subsequent erasure from the data set or in positioning the cursor exactly on a point previously moved to or drawn.

Terminal connections of the circuit are drawn in the usual schematic manner. Data entered in DATA1 table is of the same format as the normal 30-bit Adage display word.

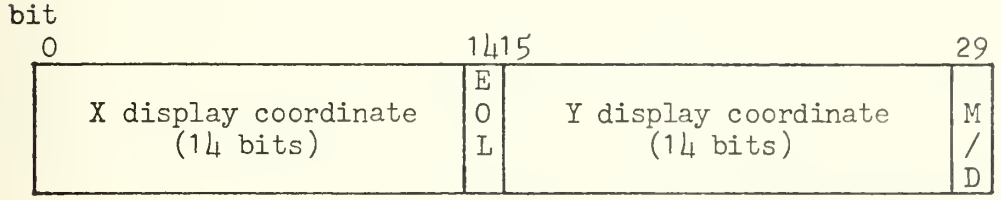

FIGURE 7

DATA1 Table Entry

The end-of-list bit (EOL) is present to one in all table entries. The remaining bits in the DATA1 table are zeroed. When the user enters a "move" or "draw" command via the function switches in the CONNECT mode the display coordinates of the cursor are stored in the next DATA1 entry. The EOL bit is cleared and the move-draw bit (abbreviated M/D above) is set if the command was a "draw". When this information is displayed, the presence of an end-of-list bit following the last DATA1 entry is assured. The DATA1 data set thus is drawn as one contiguous set of display commands.

3. Analysis Mode

The program ANALR performs analysis of the drawn circuit in two ways. Upon entering ANALR for the first time, the display tables of circuit elements created in the DEFINE mode and circuit connections entered in CONNECT are examined to determine the topology of the network.

This is accomplished by creating a bipartite graph with circuit element inputs and outputs considered as the vertices of the graph and pointers representing edges. The results of this analysis are stored in

other memory tables to be used subsequently in calculating circuit response to specified inputs.

a. Circuit Analysis

The topological analysis rests on this assumption: Any circuit element (or terminal node) which is connected to another element will have a line (or lines) representing this connection in the data set created in CONCT.

Thus, the first analysis task is to determine the end points of the connections entered. This is done by searching the DATA1 table for an instance of a "move" entry. The assumption is made that this "move" entry indicates the user is about to draw a sequence of lines representing a circuit connection. The significance of the "move" entry is that the X,Y position of the "move" will be the location of the initial point of the connection. Hence the X,Y position identified as a "move" is placed into a table called RAWBK. The succeeding DATA1 entries are searched until the next "move" entry is found indicating a new connection sequence has begun. The X,Y position of the immediately preceeding entry in the table is then recorded as the termination of the connection established by the original "move". This X,Y position is then entered into the next position in the RAWBK table. The above process continues until the entire DATA1 table has been searched. Special care is exercised to ensure that extraneous moves inadvertently entered by the user, or erased entries do not affect the extraction of topologically correct line end points.

DATA1	bit	bit
0	14	29

display X1	0	display Y1	0
" X2	0	" Y2	1
" X3	0	" Y3	1
display X4	0	display Y4	1
display X5	0	display Y5	0
. .	0	. .	1
. .	0	. .	1
. .	0	. .	1
display Xn	0	display Yn	1
0 0 0 0		0 0 0 0	
0 0 0 0		0 0 0 0	

RAWBK	bit	bit
0	14	29

display X1	0	display Y1	0
display X4	0	display Y4	1
display X5	0	display Y5	0
display Xn	0	display Yn	1
0 0 0 0		0 0 0 0	

FIGURE 8

Relationship of DATA1 and RAWBK Data Tables

The format of the entries in the RAWBK table is exactly the same as that of DATA1. The difference is that the RAWBK table contains only those entries in DATA1 which represent the end points of connections. Thus the RAWBK table is a compression of the topological information contained in the DATA1 display table.

Once the RAWBK table has been built, the basic information required for topological analysis of the circuit is present. The actual construction of the bipartite graph begins by establishing a circuit scanning loop. In this loop, each column of the grid is searched proceeding from the left to the right side of the CRT. This is done assigning an X coordinate value to the variable "level" which will be greater than any X value of a circuit termination point connected to a circuit element in that column of the grid. However, the "level" value is less than the X value of an entry in the next grid column.

All termination points entered in RAWBK which have X values to the left of this level will be mapped into one of three tables. If the display coordinates of a termination point fall within the limits

which allow a point to be connected to a gate, a determination is made whether this point should be considered as an input or output to this element. This decision is based on the side of the gate to which the point is connected. If a RAWBK entry falls within the X,Y constraints and is an input, a corresponding entry is made in the BAND (Basic ANalysis Data) table. Outputs are similarly entered into the BANDO (Basic ANalysis Data, Outputs).

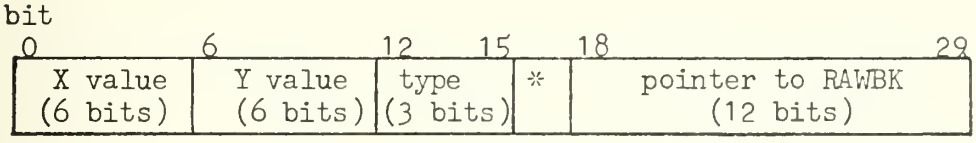

FIGURE 9

BAND Table Entry

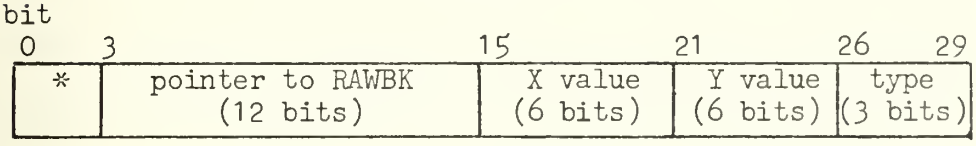

FIGURE 10

BANDO Table Entry

The field labeled * above represents a three bit value corresponding to the height of this input (or output) on the gate. This information must be extracted for use in the analysis of sequential elements.

Any RAWBK entry whose X value is to the left of the level value but does not map into a circuit element is placed in the unresolved (URD) table. (Note that all circuit inputs and outputs will generate an URD entry as one of the end points of each of these lines is not connected to a gate). The form of the unresolved data (URD) table is the same as that of RAWBK: An X,Y display coordinate value. The coordinates entered into Inputs/Outputs table (BAND/BANDO) are compressed to allow room in

the word for the gate type and connection pointer determined by this phase of ANALR. The X,Y coordinate values entered are the two most significant (Octal) digits of the display coordinate of the circuit element entries in the TBLO table. This was done because the X,Y values of the defined entries in TBLO are fixed by grid position whereas the RAWBK X (and Y) entries can vary significantly depending on the width (or height) of the circuit element. Four digits representing X,Y position together with one digit representing the type of gate are thus extracted from the TBLO entry and packed into one half word of the Inputs/Outputs table. The pointer half word contains the RAWBK address of the other end point of this connection. To make explicit the bipartite graph implication of the pointer, a BAND entry has its pointer in the lower half word (telling where this input goes) and BANDO entry pointers are in the upper half word (telling where this output comes from).

In addition to the URD (Unresolved) table, which contains the display coordinates of inputs and outputs in a full 30 bit word, the unresolved entries in RAWBK are used to create another table called "Un-Resolved Inputs/outputs" (URI). Each entry in the URI table consists of two pointers. The pointer in the top half word designates the display coordinates of this unresolved entry (i.e. a pointer into the URD table). The lower half word is a pointer to the memory location in the ANSW (answers) table reserved for the actual binary value of the input (or output) which will be specified (or calculated) later in the program. This table is required to enable user specified inputs to be placed into the proper ANSW (answer) location. Complete topological analysis is thus attained by repeating the above procedure for each grid column.

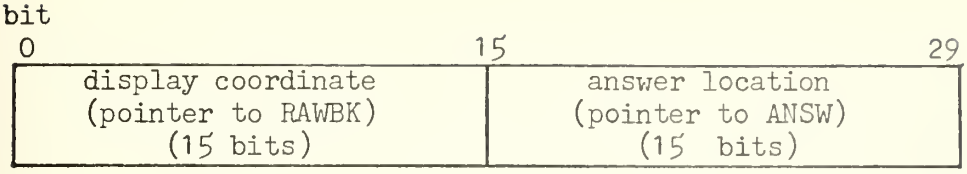

FIGURE 11

URI Table Entry

b. User Input

The entire connection analysis outlined above is transparent to the user. After its completion, ANALR (Analysis Program) automatically enters a specification phase. Two additional kinds of information are needed before circuit response can be calculated. The first of these is the labeling of terminal nodes of the circuit with up to 3 alphanumeric characters to enable a logic equation representing the circuit to be constructed. Once nodes have been labeled, actual binary inputs are accepted for each input preparatory to calculating circuit response. The node labeling phase is especially important because this is where the user will be notified of an improper circuit connection if one exists. Any entry (not a circuit input or output) which was not properly connected by CONCT will be pointed out for labeling. Hence, if the program asks for a label where the user can see that one is not required, the circuit must be re-connected.

The user is allowed to label the circuit with up to three alphanumeric characters at each circuit input and output. These characters are accepted and processed by ANALR (Analysis Program) and converted into display form for subsequent drawing on the cathode ray tube by LIL (main display mode). In order to do this, a foreground/background type of operation is set up between LIL and ANALR which allows

the continual display of already entered circuit information while waiting for teletype inputs of alphanumerics for subsequent display. The binary values input to the circuit are then accepted and processed in a similar manner. All character display information is stored in one of two tables: CTAB for label display data and NTAB for value display data. The structure of each of the tables is exactly the same.

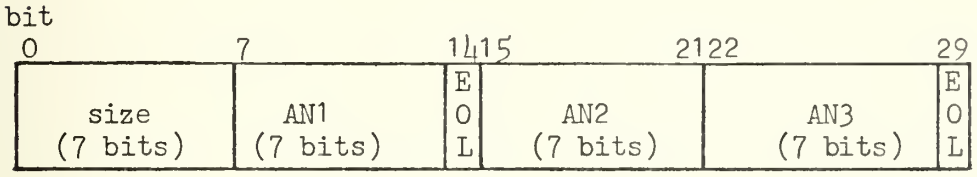

FIGURE 12

CTAB/NTAB Table Entry

The abbreviation AN above is for Alphanumeric character field, capable of storing one ASCII (American Standard Code for Information Interchange) code display character.

c. Circuit Response

The response of the circuit to specified inputs is then determined. Another table called ANSW is reserved in memory for holding the binary numbers input, and output values at any point in the circuit. Data representing both is indexed into this table by the last two digits of the RAWBK pointer in the corresponding BAND or BANDO entry, thus guaranteeing a unique storage address for each input and output value.

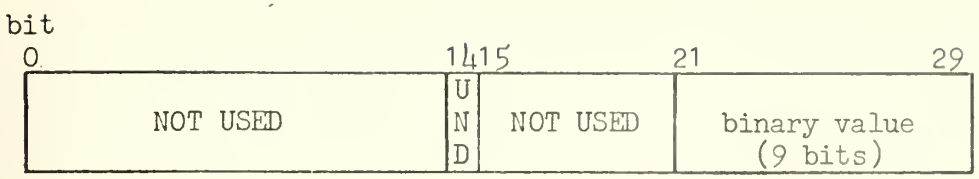

FIGURE 13

ANSW Table Entry

Another table reserved in memory is FFDT (Flip-Flop Data Table). This table is actually built upon exit from the DEFINE mode. One entry is made in this table for each flip-flop entered in DEFINE. The format of the entries is the same as the standard Adage display word: A 14 bit display X value, a 14 bit display Y value and end-of-list bits in bits 14 and 29 of the display word. The EOL bit in bit position 14 is used as an undefined flag (as in ANSW above). The EOL at bit position 29 contains the current state of the flip-flop: (set-1, reset-0). The undefined flag will be set when the state of the flip-flop cannot be calculated from current inputs.

Actual response calculation is done using a memory stack. The STACK is loaded from the top down with the binary values from ANSW which represent the inputs to the gate being simulated. A subroutine call is made to one of six subroutines to simulate the response of the circuit element to these inputs. The subroutine call is based on the gate type [Table 1].

The subroutine operates on all the stack values and places the binary result on top of the stack. Upon return to the main analysis program, this value is stored into the appropriate memory location in ANSW (answers). If the circuit element being simulated is a flip-flop some additional work must be done. Before the call to the appropriate

simulation subroutine the last state of the flip-flop must be obtained from the FFDT (Flip-Flop Data Table). This is stored at the location STACK-1 and is used to calculate the response of the flip-flop to the current input based on the last state. Upon return from the subroutine the current state of the flip-flop is entered into the appropriate FFDT location.

Circuit response calculation is carried out from left to right according to the following algorithm.

1) Locate the first (or next) circuit element in the BAND table.
2) Find the binary value in ANSW (specified by this BAND entry) and load this value into the stack.
3) Find all other inputs to this gate in BAND and load their values into the stack.
4) Calculate response of this gate by a jump to the appropriate subroutine as specified by type of gate.
5) Locate each instance of this gate in the output (BANDO) table and stuff the calculated value into the ANSW address specified by this entry.
6) If this is the last circuit element stop, if not go to step 1.

Once the response of the complete circuit has been calculated, the answers must be displayed to the user. This is done by converting the binary value in each location in ANSW corresponding to a circuit output into display form and entering it into the number display table.

After displaying the results of the calculation ANALR returns to the "INPUT VALUES" mode to allow the circuit to be tested with other input values. A new circuit may be constructed (or the current one modified) by leaving the analysis mode via the "MODE EXIT" function switch.

IV. PROGRAM EVALUATION

A. SAMPLE TERMINAL SESSION

The following computer session is documented as an example of the abilities of the program to design logic circuits. The objective of the session was to design a half-adder according to the logic expression $\bar{A}.B + A.\bar{B}$, the exclusive-or representation of a half-adder circuit. All of the following photographs were taken from the CRT. Upon program initiation the main display mode was entered resulting in the following display:

FIGURE 14

Initial Command Presentation

Selection of INTRODUCTION would display to the user a comprehensive explanation of the program and how to use it.

In this example case the DEFINE mode was selected with the light pen and the basic circuit elements of the half-adder were entered. It was desired to design the half-adder according to the above mentioned exclusive-or representation, hence two inverters, two AND gates and an OR gate were entered as shown by the following display in the DEFINE mode:

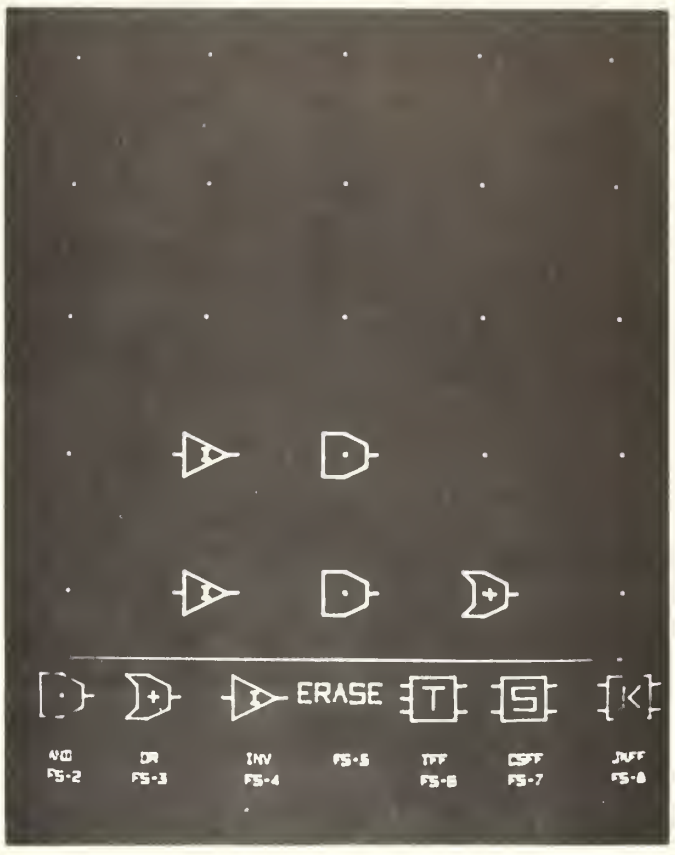

FIGURE 15

DEFINE Mode Display

After exiting the DEFINE mode (which created the above display) via the function switches the CONNECT mode was selected from the command menu of the main display mode. Connections were then established by moving the cursor with the appropriate function switches. A typical CONNECT view is shown in the following scope photograph:

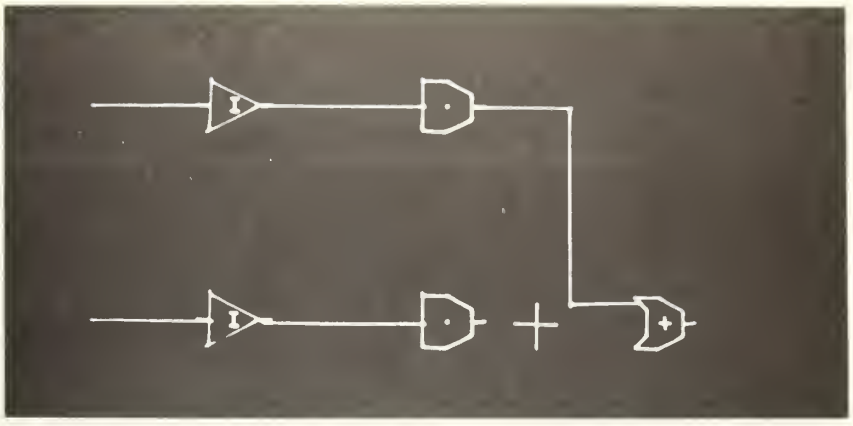

FIGURE 16

CONNECT Mode Display

The circuit was then completely connected and ANALYZE was selected after exit from the CONNECT mode to the main display mode. Initial selection of this mode causes the program to extract the topological information contained in the circuit and build the appropriate tables as discussed above. The following display signifies to the user that the program is ready for labeling of the circuit terminals and the subsequent specification of binary input values:

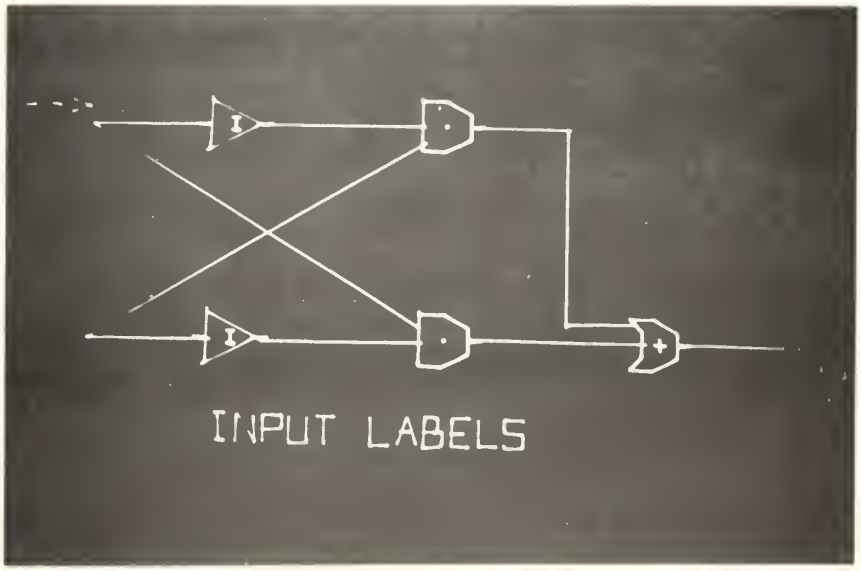

FIGURE 17

ANALYSIS Mode--Alphanumeric Input Phase

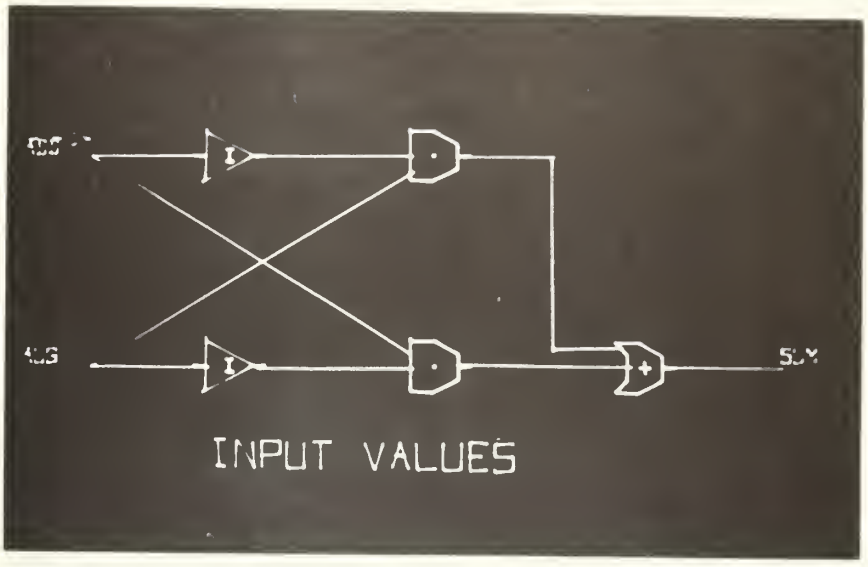

FIGURE 18

ANALYSIS Mode--Numeric Input Phase

As previously mentioned any circuit element whose connection is not properly specified upon initial entry into this phase of the program will be pointed out for labeling. When the labeling process is completed the program automatically enters the response calculation mode, wherein the user's binary teletype inputs are accepted and processed for each circuit input. The appropriate routines are called to simulate the circuit and the result of the response calculation is converted into display form and is shown for this particular circuit in the following picture:

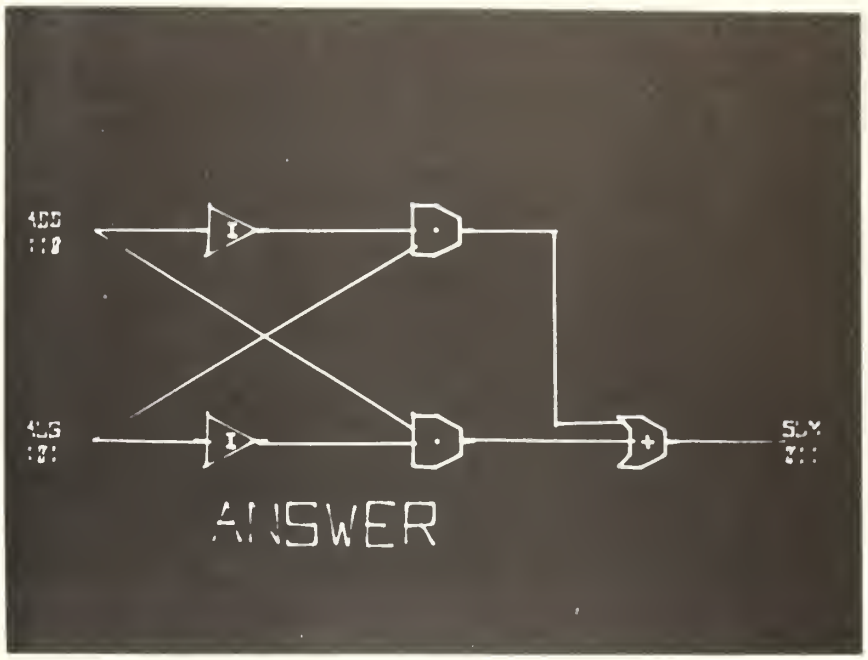

FIGURE 19

ANALYSIS Mode--Display of Response

At this point the user can go back to the main program via "mode exit" function switch or can retest the circuit with new binary inputs by depressing function switch five ("reset"). This function switch will cause the program to return to the "INPUT VALUES" display shown previously.

The following sequence of CRT photographs shows the response of a set-toggle-reset flip-flop to various binary inputs. The flip-flop is reset initially as indicated by the zero value displayed at its output terminals.

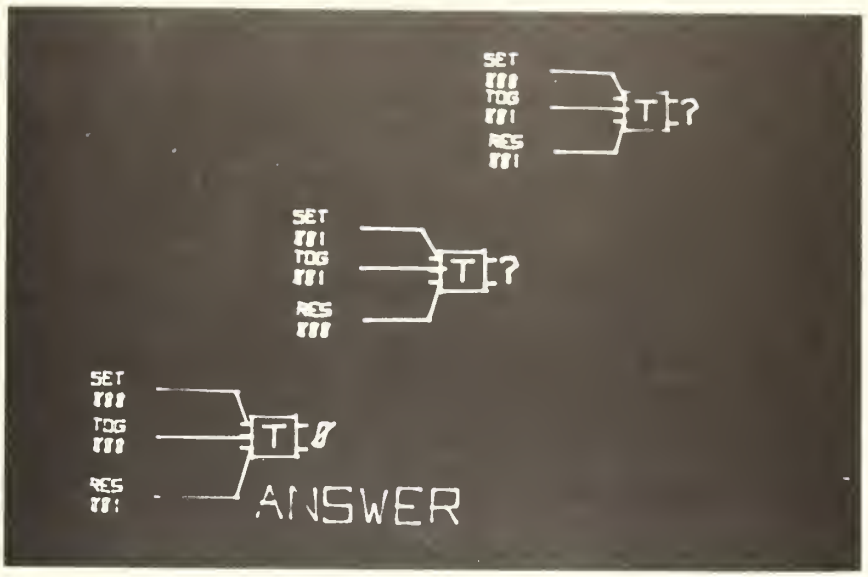

FIGURE 20

Flip-Flop Analysis(1)

A pulse applied to the set terminal causes the flip-flop to change to the "set" state (Lower left flip-flop).

Two subsequent toggle inputs change the flip-flop's value appropriately (Middle and upper right flip-flops).

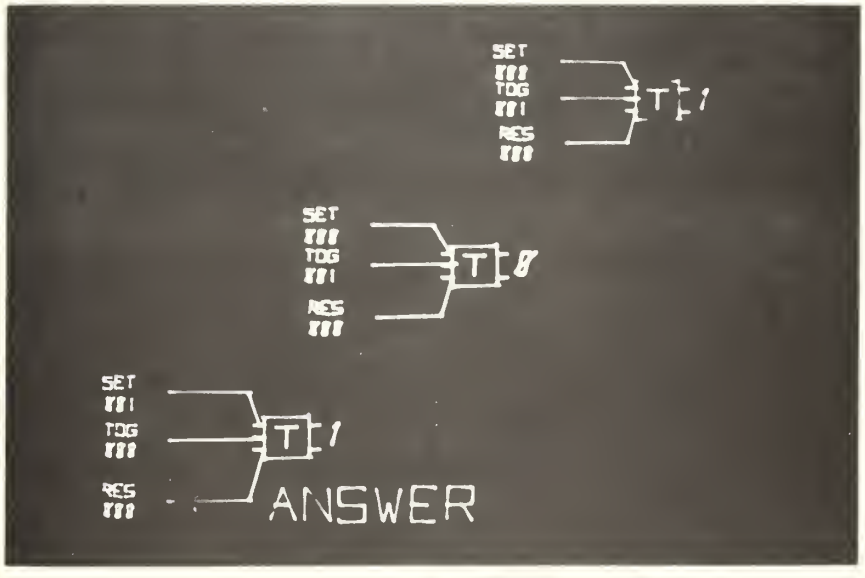

FIGURE 21

Flip-Flop Analysis(2)

A reset pulse clears the flip-flop to zero (Lower left flip-flop).

Finally the flip-flop's response to an illegal combination of inputs is shown.

B. RESULTS

At the current stage of program development the INTERACTIVE LOGIC LABORATORY has demonstrated significant ability to enable the user to create and analyze basic logic circuits. The program has not yet been used by beginning students in logical design. However, several features of the implementation should allow its eventual use as an adjunct to the classroom instruction received in the "Logical Design of Digital Computers" course (CS-3200) taught at the Naval Postgraduate School.

1) Although no generally accepted set of symbols exists for representing logic elements, the symbols used are universally identifiable since the specified operation is shown in the logic display symbol.
2) The sequential elements (flip-flops) are not standard symbols, but are quite recognizably presented.
3) The degree of student control achieved by the implementation allows the student to proceed at his own rate in the design of basic logic circuits.
4) The level of description of the INTRODUCTION mode is sufficiently comprehensive to allow most students to use the program with no assistance. Additionally, the fact that this mode does not have to be selected and only part of the instructions may be reviewed does not subject the student to tedious repetition of instructions as he gains proficiency in the use of the program.

Implementation of sequential circuits has met with less success. At the current stage of development the INTERACTIVE LOGIC LABORATORY can not be used for the design of sequential circuits. However, individual flip-flops are simulated correctly thus allowing the student to observe the response of individual elements to various inputs.

C. EXTENSIONS

Two basic inadequacies exist in the program. First of all, the bipartite graph representation does not allow any feedback loops to be present in the circuit. Since this is not an uncommon occurrence in logic circuits, the program should be modified to allow the output from a circuit element to be delayed and re-input to the same element. Secondly, the program as currently implemented does not allow for recursive constructs, hence only circuits that will fit on the 5 by 5 grid may be constructed. It is felt that the bipartite data structure is general enough to handle larger circuits, wherein a previously analyzed circuit is reduced to be considered as another primitive element.

Another area where extension of the current program is required is in the saving of a user's circuit design efforts and hard copy output. The saving of circuits could be done by punching out the display tables on paper tape. Then at a subsequent computer session the DATA1 and TBLO tables could be read into the teletype unit allowing the user to continue where he previously stopped. Hard copy output which is intelligible to the user is harder to obtain with the AGT-10, but the cheaper graphics terminals required for more general implementation of this program have provisions for hard copy output.

D. RECOMMENDATIONS AND CONCLUSIONS

It is felt that INTERACTIVE LOGIC LABORATORY has shown the feasibility of using a computer graphics terminal to demonstrate the basic concepts of logical design. One of the severe drawbacks associated with this particular implementation is the high cost of the Adage Graphics Terminal. While this could be justified by the research intent of this program, any extension of the concepts of this program into general classroom use will necessitate a lower cost graphics terminal.

Currently available storage tube graphics terminals meet these cost requirements. A classroom installation composed of individual cathode ray tubes for each student tied to a central computer capable of responding to all users in a time-shared mode would be ideal for the implementation of this logic demonstrator. Such a classroom computer installation is now available and the implementation of the INTERACTIVE LOGIC LABORATORY on this equipment would indeed provide a meaningful laboratory for experimentation in basic logical design.

APPENDIX A

Truth Table for 3-input gates

A	B	C	AND A.B.C	OR A+B+C
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth Table for clear-set flip-flop

SET	RESET	INITIAL STATE	FINAL STATE
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	? (undefined)
1	1	1	? (undefined)

Truth Table for clear-set-toggle flip-flop

SET	TOGGLE	RESET	INITIAL STATE	FINAL STATE
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	?
0	1	1	1	?
1	0	0	0	1
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	?
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Truth Table of J-K flip-flop

J	K	INITIAL STATE	FINAL STATE
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

```
EXPUNGE
TITLE L0GMM                    [THIS PR0GRAM IMPLEMENTS THE DEFINE M0DE
ENTRY STAR,DADD,TBL0,FCLER,D0RD,DINVD,D0TD,FLG1,ADDD,T1,TCNT,RETUR,E0PFL,TXT
ENTRY DUNN,FFTD,FFSD,FFKD,FFDT

MACR01         P(A1,A2,A3,A4,A5)
     A1VBVK + A2VB + A5VBVK + A3VBVK + A4VB +0
ENDM

STAR:          N0@P
               ARMD            SAVAR
               ARX0'F                         [CLEAR S0ME FLAGS
               ARMD            CFLG
               ARMD            $ANFLG
               ARMD'0          $DF1FG
               MDAR'N          FSET           [SET UP F/C C0UNT
               MDAE            C1
               ARMD            FS
               MD10            SAVG           [2D N0RMAL,SC0PE AND FC 0N
               MDAR'F          FCLER          [L0AD CL0CK PIV0T
               ARMD            77755
               MDAR'F          E0LER          [L0AD E0L PIV0T
               ARMD            77757
               MDAR'F          LPHAN          [L0AD LP PIV0T
               ARMD            77760
               MDAR            TBL0           [SKIP TBL0 INITIAL IF
               ARAR'A'L;       77777          [THIS IS FIRST ENTRY
               JPLS            .+3
               MDAR'F          TBL0
               ARMD            TBL0           [C0NTENT 0F T0P ENTRY IN
                                              [TBL0 P0INTS T0 LAST
               MDAR'F          ETBL0          [USED ENTRY
               ARMD            ETBL0          [LAST TABLE ENTRY

               MDAR            SAVAR
               JUMP            .              [HANG WAIT F0R F/C TICK
```

```
EOLER:      NOOP                        [END OF LIST HANDLER
            ARMD        SAVDR
            MDAR'X      LFLG            [SET IMAGE DONE FLAG
            MDAR        SAVDR
            JUMP'I      EOLER           [RETURN TO CALL
FCLER:      NOOP                        [FRAME CLOCK HANDLER
            ARMD        SAVDR
            MD10'A'L)   -1000JH         [TURN OFF FRAME CLOCK
            MDAR'X      FCNT            [INCREMENT FC COUNT
            MDAR        LFLG            [CHECK IMAGE DONE FLAG
            JPLS        .+2
            JUMP        FCL1            [JUMP TO END IF NOT DONE
LUPE:       JPSR        $FIN
                        ATBL
            MDAR        CFLG            [CHECK DEFINE EXIT
            JPAN        BFFDT           [EXIT TO LIL
            MDAR        FS              [COMPARE REQUIRED TICKS
            MDAS        FCNT            [WITH ACTUAL TICKS
            JPAN        FCL1
            MDAR        FCLER
            ARMD        RETUR
            MDAR'F      RETUR+1         [GO TO DRAW ROUTINES
            ARMD        FCLER           [IF GREATER THAN 3 FC
            ARXO'F
            ARMD        FCNT
FCL1:       MD10'O'L)   1000JH
            MDAR        SAVDR           [CLEAR F/C INTERRUPT
            JUMP'I      FCLER           [AND RETURN TO CALL
RETUR:      NOOP
            ARMD        SAVDR
            MDAR        LFLG
            JPLS        .+2
            JUMP        .-2
            MD10'A'L)   -1000JH         [TURN OFF FC
            MDAR        TCNT            [SUBTRACT GET COUNTER FROM
            MDAE'N      GCNT            [TCNT
```

```
                JPAN        DD          [JUMPS TO DOT DRAW IF TABLE EMPTY
                MDAR'X      GCNT
                MDAR'F      TBL0
                MDAE        GCNT        [GET NEXT ADDRESS IN TABLE
                ARMD        T1
                MDAR'I      T1          [LOAD CONTENTS OF TBL0 PLUS GCNT
                MDAR'A      C3          [MASK ALL BUT ID BITS
                MDAE'N      C1          [PICK PROPER SUBROUTINE
                JPAN        ADDP
                MDAE'N      C1
                JPAN        SRP
                MDAE'N      C1
                JPAN        INVP
                MDAE'N      C1
                JPAN        FFT1
                MDAE'N      C1
                JPAN        FFS2
                MDAE'N      C1
                JPAN        FFK3
ADDP:           MDAR'X      FLG1        [SET ADD FLAG
                MDAR'I      T1          [LOAD DX AND DY INTO AR
                MDAR'A      C2          [MASK ID BITS
                ARMD        DADD
                JPSR        DRW
                ARX0'F
                ARMD        FLG1        [RESET FLG1
                JUMP        RETUR+2
SRP:            MDAR'X      FLG1        [SET FLG1 TO 2
                MDAE        C1
                ARMD        FLG1
                MDAR'I      T1
                MDAR'A      C2
                ARMD        DORD        [OR DATA
                JPSR        DRW
                ARX0'F
                ARMD        FLG1
```

```
        JUMP        RETUR+2
INVP:   MDAR'X      FLG1
        MDAE        C4          [SET FLG1 TO 3
        ARMD        FLG1
        MDAR'I      T1
        MDAR'A      C2
        ARMD        DINVD       [INV DATA
        JPSR        DRW
        ARX0'F
        ARMD        FLG1
        JUMP        RETUR+2
FFT1:   MDAR'X      FLG1
        MDAE        C6
        ARMD        FLG1        [SET FLG1 FOR FFT
        MDAR'I      T1
        MDAR'A      C2
        ARMD        FFTD        [FFT DATA
        JPSR        DRW
        ARX0'F
        ARMD        FLG1
        JUMP        RETUR+2
FFS2:   MDAR'X      FLG1
        MDAE        C6
        MDAE        C1
        ARMD        FLG1        [SET FLG1 FOR FFS
        MDAR'I      T1
        MDAR'A      C2
        ARMD        FFSD        [FFS DATA
        JPSR        DRW
        ARX0'F
        ARMD        FLG1
        JUMP        RETUR+2
FFK3:   MDAR'X      FLG1
        MDAE        C4
        MDAE        C6
        ARMD        FLG1        [SET FLG1 FOR FFK
```

```
                MDAR'I          T1
                MDAR'A          C2
                ARMD            FFKD            [FFK DATA
                JPSR            DRW
                ARXØ'F
                ARMD            FLG1
                JUMP            RETUR+2
DD:             ARXØ'F
                ARMD            GCNT            [RESET GET CØUNTER TØ 0
                JPSR            DTDRW           [DRAW DØTS
DM:             MDAR            D1
                ARMD            DADD
                MDAR'X          FLG1            [THIS SECTIØN DRAWS
                JPSR            DRW
                MDAR            LFLG
                JPLS            .+2
                JUMP            .-2
                MDAR'X          FLG1            [THE MENU.
                MDAR            D2
                ARMD            DØRD
                JPSR            DRW
                MDAR            LFLG
                JPLS            .+2
                JJMP            .-2
                MDAR'X          FLG1
                MDAR            D3
                ARMD            DINVD
                JPSR            DRW
                MDAR            LFLG
                JPLS            .+2
                JJMP            .-2
                MDAR'X          FLG1
                MDAR            D4
                ARMD            FFTD
                JPSR            DRW
                MDAR            LFLG
```

```
                JPLS            .+2             [CONTINUE MENU DRAW
                JUMP            .-2
                MDAR'X          FLG1
                MDAR            D5
                ARMD            FFSD
                JPSR            DRW
                MDAR            LFLG
                JPLS            .+2
                JUMP            .-2
                MDAR'X          FLG1
                MDAR            D6
                ARMD            FFKD
                JPSR            DRW
                MDAR            LFLG
                JPLS            .+2
                JUMP            .-2
                ARX0'F
                ARMD            FLG1
TXTW:           ARX0'F
                ARMD            LFLG
                MD10'A'L;       17777JH         [TURN OFF VECTOR GEN
                MDIC'A'L;       77763           [SET IC BITS 26,27
                MDAR'F          TXT-1
                ARMD            77735
                MDAR'F          TEOLR
                ARMD            77736
                MDIC'0'L;       10              [TURN ON CHAR GEN
                MDAR            LFLG
                JPLS            .+2
                JUMP            .-2
                MDIC'A'L;       77763           [RESET IC BITS 26,27
INDRA:          MDAR            XFLG
                JPAN            .+2
                JUMP            .+2
                JPSR            INDRW
                MD10'0'L;       1000JH
```

```
                MDAR            SAVDR
                JUMP'I          RETUR
TEBLR:          NBBP
                ARMD            SAVTX
                MDIC'A'L;       -10             [TURN BFF CHAR GEN
                MD10'B'L;       60000JH         [TURN BN THE VECTBR GEN
                MDAR'X          LFLG
                MDAR            SAVTX
                JUMP'I          TEBLR
LPHAN:          NBBP
                ARMD            SAVB
                MD10'A'L;       -20JH           [TURN BFF LP
                MDAR'I          77756           [LBAD DXDY PBSIT BEING DRAWN INTB AR
                MDAR'A          C1              [MASK ALL BUT LAST BIT-M/D BIT
                JPLS            .+7             [JUMP IF DRAW BIT SET
                MDAR            77756
                MDAE'N          C1
                ARMD            C5
                MDAR'I          C5              [LBAD CBNTENTS BF DXDY PBSIT INTB AR
                ARMD            DXDY            [PUT PBSIT BF SELECTED DBT IN DXDY
                JUMP            .+3
                MDAR'I          77756
                ARMD            DXDY
                MDAR'H          SCALD
                MPYU            DXDY
                NBBP
                MDAR'A          AA1
                ARMD            TEM
                MDAR'H          SCALD
                MPYL            DXDY
                NBBP
                ARRS            17
                NBBP
                MDAR'A          A2
                MDAE            TEM
                MDAE            DBTD
```

```
                ARMD            DXDY
                MD10'B'L;       20JH            [TURN BN LP
                MDAR            FS
                ARMD            XFLG
ZF:             MDAR            SAVB
                JUMP'I          LPHAN
ATBL:           0               CNCT            [TABLE BF DEFINE FNS
                0               ADDD            [FNS-2
                0               BRR             [FNS-3
                0               INV             [FNS-4
                0               ERAS            [FNS-5
                0               FFT             [FNS-6
                0               FFS             [FNS-7
                0               FFK             [FNS-8
                REPEAT          10.             [FNS-1 IS MBDE EXIT
                0               LNULL
                ENDR
ADDD:           JUMP            .               [FBLLBWING RBUTINES PICK
                ARMD            SAVB            [THE PRBPER BITS VALUE
                MDAR            XFLG            [FBR TYPE BF GATE
                JPAN            .+3
                MDAR            SAVB
                MDIR            ADDD
                MDAR            C1
                ARMD            BTS
                JPSR            CBMBI
                MDAR            SAVB
                MDIR            ADDD
BRR:            JUMP            .
                ARMD            SAVB
                MDAR            XFLG
                JPAN            .+3
                MDAR            SAVB
                MDIR            BRR
                MDAR            C4
                ARMD            BTS
```

```
                JPSR            COMBI           [ROUTINES TO PICK
                MDAR            SAV0            [BITS (CONTD.)
                MDIR            0RR
INV:            JUMP            .
                ARMD            SAV0
                MDAR            XFLG
                JPAN            .+3
                MDAR            SAV0
                MDIR            INV
                MDAR            C6
                ARMD            BTS
                JPSR            COMBI
                MDAR            SAV0
                MDIR            INV
FFT:            JUMP            .
                ARMD            SAV0
                MDAR            XFLG
                JPAN            .+3
                MDAR            SAV0
                MDIR            FFT
                MDAR            C6
                MDAE            C1
                ARMD            BTS
                JPSR            COMBI
                MDAR            SAV0
                MDIR            FFT
FFS:            JUMP            .
                ARMD            SAV0
                MDAR            XFLG
                JPAN            .+3
                MDAR            SAV0
                MDIR            FFS
                MDAR            C6
                MDAE            C4
                ARMD            BTS
                JPSR            COMBI
```

```
                MDAR            SAV0            [R0UTINES T0 PICK BITS
                MDIR            FFS             [C0NTINUED
FFK:            JUMP            .
                ARMD            SAV0
                MDAR            XFLG
                JPAN            .+3
                MDAR            SAV0
                MDIR            FFK
                MDAR            C6
                MDAE            C6
                ARMD            BTS
                JPSR            C0MBI
                MDAR            SAV0
                MDIR            FFK
LNULL:          JUMP            .               [D0-N0THING R0UTINE
                MDIR            LNULL           [F0R FUNC. SWITCHES
ERAS:           JUMP            .               [R0UTINE T0 ERASE A
                ARMD            SAV0            [SELECTED ELEMENT
                MDAR            XFLG
                JPAN            .+3
                MDAR            SAV0
                MDIR            ERAS
                ARX0'F
                ARMD            ECNT
                MDAR            DXDY
                MDAR'A          C2              [MASK ID BITS
                ARMD            DXDYE
                MDAR'F          ETBL0-2         [L0AD ADDRESS 0F LAST
                ARMD            ZZE             [TBL0 L0CATI0N INT0 ZZE
                MDAR'F          TBL0            [L0AD STARTING ADDRESS 0F
                ARMD            ZZZ             [TBL0 INT0 ZZZ
                MDAR'X          ECNT
                MDX0            C26
                JPLS            .+3
                MDAR            SAV0
                MDIR            ERAS
```

```
                MDAR'X      ZZZ         [INDEX ZZZ T0 NEXT TBL0 ENTRY
                MDAR'I      ZZZ         [L0AD C0NTENTS 0F ZZZ T0 AR
                MDAR'A      C2          [MASK ID BITS
                MDX0        DXDYE       [AND C0MPARE WITH DXDYE
                JPLS        •-9•
                ARAR'F'H
                JPLS        •-11•
                MDAR        ZZE         [CHECK IF LAST ENTRY IN TBL0,
                MDX0        ZZZ         [IF N0T JUMP AHEAD
                JPLS        •+14•       [14 INST
                MDAR        TBL0        [IF LAST ENTRY
                MDAE'N      C1          [DECREMENT TBL0 PMINTER AND
                ARMD        TBL0        [TCNT C0UNTER
                MDAR        TCNT
                MDAS'N      C1
                JPAN        •+2
                JUMP        •+2
                ARX0'F
                ARMD        TCNT
                MDAR        C1
                ARMD        XFLG
                MDAR        SAV0
                MDIR        ERAS        [REST0RE AR AND RETURN
                MDAR        ZZZ
                MDAE        C1
                ARMD        T1
                MDAR'I      T1
                ARMD'I      ZZZ
                MDAR'X      ZZZ         [ZZZ THEN STEP
                JUMP        •-22•
CNCT:           JUMP        •           [THIS R0UTINE SETS THE
                ARMD'0      CFLG        [EXIT FLAG
                MDIR        CNCT
C0MBI:          JUMP        •           [ADDS TYPE 0F GATE
                ARMD        SAVX        [INT0 L0WER DIGITS 0F
                MDAR'X      TBL0        [TBL0 TABLE
```

```
                MDAR            DXDY            [ERASE, CONTINUED
                MDAR'A          C2
                MDAE            BTS
                ARMD'I          TBL0
                MDAR'X          TCNT
                MDAR            TBL0
                MDAE            C1
                MDX0            ETBL0
                JPLS            .+2
                JUMP            E0P
                MDAR            C4
                ARMD            XFLG
                MDAR            SAVX
                MDIR            C0MBI
INDRW:          N90P                            [THIS R0UTINE DRAWS
                ARMD            ASAV            [THE BASIC SQUARE
                MD06            INT
                MD11            SCLS
                MDAR            DXDY
                MDAR'A          C2
                ARMD            SQR
                MD07            SQR
                MDAR'L
                MD05            SQR
                ARMD            77756
                ARX0'F
                ARMD            LFLG
                MD05            SQR
                MDAR            ASAV
                JUMP'I          INDRW
DRW:            N80P                            [THIS R0UTINE DRAWS
                ARMD            ASAV            [THE PR0PER GATE
                MD11            SCAL            [SET SCALE
                MD06            INT             [SET INTENSITY
                ARX0'F
                ARMD            LFLG
```

```
                MDAR            FLG1            [TURN OFF IMAGE DONE FLG
                MDAE'N          C1
                JPAN            A1              [JUMP TO DRAW ADDER
                MDAE'N          C1
                JPAN            O1              [JUMP TO DRAW ORRER
                MDAE'N          C1
                JPAN            I1              [JUMP TO DRAW INVERTER
                MDAE'N          C1
                JPAN            FF1             [JUMP TO DRAW TRIG FF
                MDAE'N          C1
                JPAN            FF2             [JUMP TO DRAW CS FF
                MDAE'N          FF3
                JPAN            FF3             [JUMP TO DRAW JK FF
A1:             MDO7            DADD
                MDAR'L
                MDO5            DADD
                ARMD            77756
                MDO5            DADD            [STARTS DRAWING
                JUMP            ENDRW           [DRAW AND GATE
O1:             MDO7            DORD
                MDAR'L
                MDO5            DORD
                ARMD            77756
                MDO5            DORD            [DRAW OR GATE
                JUMP            ENDRW
I1:             MDO7            DINVD
                MDAR'L
                MDO5            DINVD
                ARMD            77756
                MDO5            DINVD           [DRAW INVERTER
                JUMP            ENDRW
FF1:            MDO7            FFTD
                MDAR'L
                MDO5            FFTD
                ARMD            77756
                MDO5            FFTD            [DRAW TOGGLE F-F
```

```
                JUMP            ENDRW           [CONTINUE TO DRAW
FF2:            MD07            FFSD
                MDAR'L
                MD05            FFSD
                ARMD            77756
                MD05            FFSD            [DRAW SET-CLEAR F-F
                JUMP            ENDRW
FF3:            MD07            FFKD
                MDAR'L
                MD05            FFKD
                ARMD            77756
                MD05            FFKD            [DRAW J-K F-F
                JUMP            ENDRW
ENDRW:          MDAR            ASAV
                JUMP'I          DRW
BFFDT:          MDAR'F          TBL0            [THIS ROUTINE BUILDS THE
                MDAR'A          MASK1           [FLIP-FLOP DATA TABLE
                ARMD            TTEMP
                MDAR'F          FFDT+1          [SET POINTERS TO TOP
                MDAR'A          MASK1           [OF TABLE
                ARMD            DTEMP
FTRET:          MDAR'X          TTEMP           [GET NEXT TBL9 ENTRY
                MDAR            TBL0
                MDAE'L,         1
                MDX0            TTEMP           [JUMP OUT IF LAST
                JPLS            .+2             [TBL0 ENTRY
                JUMP            FTE
                MDAR'I          TTEMP
                MDAR'A'L,       7
                MDAE'N'L,       3               [SEE IF IT IS A F-F
                JPAN            FTRET           [GET NEXT IF NOT
                MDAR'I          TTEMP
                MDAR'A'L,       -100007         [CLEAR BITS 14,27,28,29
                ARMD'I          DTEMP           [STORE IN FFDT
                MDAR'X          DTEMP           [GET NEXT F-F ENTRY IN
                JUMP            FTRET           [TBL0
```

```
FTE:            MDAR            DTEMP           [FFDT BUILT
                ARMD            FFDT            [EXIT T0 LIL
                JUMP'I          EXIT
DTDRW:          N00P                            [DRAWS THE GRID D0TS
                ARMD            ASAV
                MD06            INT             [SET INTENSITY
                ARX0'F
                ARMD            LFLG            [TURN 0FF IMAGE D0NE FLG
                MD11            SCALD           [SET SCALE
                MD07            D0TD
                MD10'0'L;       20JH            [TURN 0N LP
                MDAR'L
                MD05            D0TD            [START THE DRAWING
                ARMD            77756
                MD05            D0TD            [DRAW THE D0TS
                MDAR            LFLG
                JPLS            .+2
                JUMP            .-2
                MD10'A'L;       -20JH
                MDAR            ASAV
                JUMP'I          DTDRW
LPAN:           N00P
E0P:            MDAR            C1              [THIS PR0GRAM EXIT IS N0T
                ARMD            E0PFL           [USED.
                MD10'A'L;       -1000JH
[C0NSTANTS AND DATA]
SAVAR:0
FSET:2
C1:1
FS:0
SAVG:61400JH
DXDY:7057600000
ASAV:0
LFLG:1
SAVDR:0
FCNT:0
```

```
ARSAV:0
SCAL:07770JH
INT:14000
C2:7777677770
TCNT:0
GCNT:0
FLG1:0
T1:0
C3:7
C4:2
E8PFL:0
SAV8:0
D1:6277661776
D2:6677661776
D3:7367661776
C5:0
XFLG:2
SCLS:14000JH
SAVTX:0
D4:0400061776
EXIT:$DDEF
CFLG:0
DUMN:0
SCALD:34000JH
AA1:77776JH
A2:77776
TEM:0
D5:1010061776
D6:1500061776
C6:3
BTS:0
DXDYE:0
ZZZ:0
ZZE:0
SAVX:0
ECNT:0
```

```
C26:32
MASK1:77777
TTEMP:0
DTEMP:0
FFDT:       0               0           [FLIP-FLOP DATA TABLE
            REPEAT          15.
            0               0
            ENDR
TBL0:       0                           [GATE DISPLAY TABLE
            REPEAT          26.
            0
            ENDR
ETBL0:      0
DADD:       0000000000                  [DATA TO DRAW AND GATE
            0000004000
            0400002001
            0400075777
            0000073777
            7377673777
            7377604001
            0000004001
            0400000000
            0600000001
            0000000000
            0000000001
            0000100000
DDTD:       0000003400                  [DATA TO DRAW THE DOTS
            6137616400
            6137616401
            7057616400
            7057616401
            0000016400
            0000016401
            0720016400
            0720016401
            1640016400
```

```
1640016401
1640007200
1640007201
0720007200
0720007201
0000007200
0000007201
7057607200
7057607201
6137607200
6137607201
6137600000
6137600001
7057600000
7057600001
0000000000
0000000001
0720000000
0720000001
1640000000
1640000001
1640070576
1640070577
0720070576
0720070577
0000070576
0000070577
7057670576
7057670577
6137670576
6137670577
6137661376
6137661377
7057661376
7057661377
0000061376
```

[CONTINUE DST DATA

```
                0000061377
                0720061376
                0720061377
                1640061376
                1640061377
                1640055776
                6137655777
                0000100000
DBRD:   0000000000                      [DATA TO DRAW BR GATE
                0000004000
                0400002001
                0400075777
                0000073777
                7377673777
                7577675777
                7577602001
                7377604001
                0000004001
                0400000000
                0600000001
                0100001000
                0100076777
                0200000000
                0000000001
                0000100000
DINVD:  0000000000                      [DATA TO DRAW INVERTER
                7377604000
                0400000001
                0600000001
                0400000000
                7377673777
                7377604001
                7377600000
                7177600001
                0000001000
                0000076777
```

```
                0050076777          [INVERTER DATA CONTD.
                7727676777
                0050001000
                7727601001
                0000100000
SQR:            0000000000          [BASIC SQUARE DATA
                0100001000
                0100076777
                7677676777
                7677601001
                0100001001
                0000100000
FFTD:           0000000000          [TOGGLE F-F DATA
                0400004000
                7377604001
                7377673777
                0400073777
                0400004001
                0400002300
                0600002301
                0400075476
                0600075477
                7377602300
                7177602301
                7377600000
                7177600001
                7377675476
                7177675477
                0000002000
                0000075777
                0200002000
                7577602001
                0000100000
FFSD:           0000000000          [CLEAR-SET F-F DATA
                0400004000
                7377604001
```

```
                7377673777          [CLEAR-SET F-F DATA
                0400073777          [CONTINUED
                0400004001
                0400002300
                0600002301
                0400075476
                0600075477
                7377602300
                7177602301
                7377675476
                7177675477
                0200002000
                7577602001
                7577600001
                0200000001
                0200075777
                7577675777
                0000100000
FFKD:           0000000000          [J-K FLIP-FLOP DATA
                0400004000
                7377604001
                7377673777
                0400073777
                0400004001
                0400002300
                0600002301
                0400075476
                0600075477
                7377602300
                7177602301
                7377675476
                7177675477
                7677602000
                7677675777
                0200002000
                0000000001
```

```
                0200075777                      [J-K F-F DATA C0NTINUED
                0000100000
TXT:            P(22,11,29.,13)                 [TEXT DATA F0R THE MENU
                P(102.,106,123,55)
                P(62,11,39.,106)
                P(123,55,63,11)
                P(52.,106,123,55)
                P(64,11,72.,106)
                P(123,55,66,11)
                P(82.,106,123,55)
                P(67,11,94.,106)
                P(123,55,70,0)
                P(13,100.,11,29.)
                P(101,116,104,11)
                P(40.,117,122,11)
                P(52.,111,116,126)
                P(11,72.,124,106)
                P(106,11,82.,103)
                P(123,106,106,11)
                P(94.,112,113,106)
                P(106,11,62.,106)
                P(123,55,65,13)
                P(94.,11,58.,23)
                P(105,122,101,123)
                P(105,0,0,0,1)
MD05=25000JH                    [USER DEFINED C0MMANDS
MD06=26000JH
MD07=27000JH
MD10=30000JH
MD11=31000JH
TERMINATE                       [END 0F L0GMM (DEFINE M0DE).....
```

```
EXPUNGE
TITLE C0NCT                    [THIS PR0GRAM IMPLEMENTS C0NNECT

ENTRY C0NCT,HDXY,HDXY1,DATA,DATA1,DATA2,C11,TEMP

C0NCT:      N00P
SKIP:       JUMP        .+1
            JPSR        CINIT       [TURNS 0N CURS0R
            ARX0'F
            ARMD        ECFLG       [CLEAR EXIT FLAG
            ARMD        $ANFLG
            JPSR        $FIN        [G0 CHECK FNS
                TABLE
            MDAR        ECFLG       [TIME T0 EXIT
            JPAN        $DDEF       [YES, S0 D0 IT
            MDAR'L;     2           [ELSE
            MDAE'L
            JUMP        $DDEF
            ARIR                    [EXITS T0 DDEF+2 T0 DRAW
TABLE:      0           ENDCT       [FNS TABLE--M0DE EXIT FNS-1
            REPEAT      3.
            0           CNULL
            ENDR
            0           ERASV       [FNS-5--ERASE A LINE
            REPEAT      3.
            0           CNULL
            ENDR
            40000/H     CUP         [FNS-9--CURS0R UP
            40000/H     CLEFT       [FNS-10--CURS0R LEFT
            40000/H     CRITE       [FNS-11--CURS0R RIGHT
            0           SVF         [FNS-12--SELECT VECT0R FWD
            40000/H     CDWN        [FNS-13--CURS0R D0WN
            0           MVEC        [FNS-14--M0VE
            0           DVEC        [FNS-15--DRAW
            0           CNULL
            0           CNULL
```

```
                0               SVB             [FNS-16--SELECT VECT0R BKWD


CLEFT:          JUMP            .               [M0VE CURS0R LEFT
                MDAR            HDXY
                MDAE'N          DDX
                ARMD            HDXY
                MDIR            CLEFT

CRITE:          JUMP            .               [M0VE CURS0R RIGHT
                MDAR            HDXY
                MDAE            DDX
                ARMD            HDXY
                MDIR            CRITE

CUP:            JUMP            .               [M0VE CURS0R UP
                MDAR            HDXY
                MDAS            DDY
                ARMD            HDXY
                MDIR            CUP

CDWN:           JUMP            .               [M0VE CURS0R D0WN
                MDAR            HDXY
                MDAS'N          DDY
                ARMD            HDXY
                MDIR            CDWN

MVEC:           JUMP            .               [SET M0VE VECT0R
                MDAR            HDXY
                MDAR'A          MASK1
                ARMD'X'I        TEMP
                MDAR'X          C0UNT
                MDIR            MVEC

DVEC:           JUMP            .               [SET DRAW VECT0R
```

```
                MDAR            HDXY
                MDAR'A          MASK1           [ ZERØS IN BITS 14 AND 30 ]
                MDAR'Ø          MASK2           [ ØNE IN BIT 30 ]
                ARMD'X'I        TEMP
                MDAR'X          CØUNT
                MDIR            DVEC

S7:             JUMP            0               [ TURN CURSØR ØFF ]
                MDAR'X          C1
                MDIR            S7

CINIT:          JUMP            .               [TURN CURSØR ØN
                MDAR            C2              [ALSØ SETS UP PØINTERS IN
                ARMD            C1              [DATA1 TØ INITIALIZE
                ARMD            HDXY            [CØNNECT TABLE
                MDAR'F          DATA1-1
                ARMD            TEMP
                ARMD            TEMP1
                ARMD            TEMP2
                MDAR'X'I        TEMP2
                MDAR'X          SKIP
                MDIR            CINIT

SVF:            JUMP            .               [SELECT VECTØR FØRWARD
                MDAR'X'I        TEMP1
                MDAR'A          MASK1
                ARMD            DATA2+1
                ARMD            HDXY
                MDAR'X'I        TEMP2
                ARMD            DATA2+2
                MDAR            C2
                ARMD            C11
                MDIR            SVF

ERASV:          JUMP            .               [ERASE VECTØR
                ARXØ'F
```

```
                ARMD'I          TEMP2
                MDAR'X          C11
                MDIR            ERASV

SVB:            JUMP            .               [SELECT VECT0R BACKWARD
                MDAR            TEMP1
                MDAE            D1
                ARMD            TEMP1
                MDAR'I          TEMP1
                MDAR'A          MASK1
                ARMD            DATA2+1
                ARMD            HDXY
                MDAR            TEMP2
                MDAE            D1
                ARMD            TEMP2
                MDAR'I          TEMP2
                ARMD            DATA2+2
                MDAR            C2
                ARMD            C11
                MDIR            SVB
ENDCT:          JUMP            .               [SET THE EXIT FLAG
                ARMD'0          ECFLG
                MDIR            ENDCT
CNULL:          JUMP            .               [D0-N0THING R0UTINE
                MDIR            CNULL

[C0NSTANTS AND VARIABLES]

                ECFLG:0
                C0UNT:1
                HDXY:0
                HDXY1:0
                SAVEA:0
                SCAL:37777JH
```

```
                INT:20000                       [CONSTANTS AND VARIABLES
                DDX:50√H                        [CONTINUED
                DDY:50
                MASK1:7777677776
                MASK2:0000000001
                C1:1
                C2:0
                THETA:0
                ASAV:0
                TEMP:0
                TEMP1:0
                TEMP2:0
                SAVE:0
                C11:1
                D1:-1

DATA:           0050000000                      [DATA TO DRAW THE CURSOR
                7727600001
                0000000500
                0000077277
                0000000000
                0000100000

DUMMY:          0000000000
DATA1:          0000000000                      [TABLE OF CONNECTION LINES
                0000100000
                0000100000
                0000100000
                REPEAT          60.
                0000100000
                ENDR

DATA2:          0000000000                      [TABLE TO REDRAW THE
                0000000000                      [SELECTED VECTOR
                0000000000
                0000100000
```

```
0000100000
0000100000

MD05=25000JH                    [USER DEFINED INSTRUCTIONS
MD10=30000JH
MD06=26000JH
MD07=27000JH
MD11=31000JH

TERMINATE       [END OF CONCT (CONNECT MODE)
```

```
EXPUNGE

TITLE ANALR                    [THIS PROGRAM IMPLEMENTS DEFINE MODE
ENTRY ANAL,RAWTP,RBP,RAWRT,RAWBK
ENTRY CANAL,RTCAL,LSRCH,TBSCH,CHKX,CHKY,BANAL,UNRES,DOAGN
ENTRY BAND,URD,LEVEL,STUFP,STUF,ABOV,BANDO
ENTRY TAEL,TAELR,TAZE,BKSTF,EOLPD,ENDTA,FULD
ENTRY DT1P,NDT1E,BDT1,ENDCA,ANFLG,RIT
ENTRY RESPO,COPB,NBSCH,NBSR1,NBSR2,COMP,OUTP,OTSCH,SUNP
ENTRY QPTR,NBAND,NBP,NBBOT,STACK,STKPT,STKBT,ID,OP,LNBP
ENTRY OPNS,OTPTR,ANSW,BURD,CCNT,CHAR,CFLAG,URI,URIPT
ENTRY BTURI,NOUTS,ANFNS,ANXIT,ANRST,XITFG,RSTFG,ANCHK
ENTRY CMASK

ANAL:          NOOP
               MD10!A'L;       -1000/H         [SHUT OFF THE F/C
               ARXO'F
               ARMD            XITFG           [CLEAR EXIT AND RESET
               ARMD            RSTFG           [FLAGS
               JPSR            $FIN            [GO CHECK ANALR FNS
                               ANFNS
               MDAR            XITFG           [SEE IF EXIT OR RESET
               JPAN            $DDEF           [GO TO MAIN DISPLAY MODE
               MDAR            RSTFG
               JPAN            $DR3            [GO TO SET UP FOR NUMBS AGAIN
ANCHK:         NOOP
               MDAR            ANFLG           [CHECK-SEE IF ANALYSIS NEEDS
               JPLS            AN1             [TO BE DONE FOR FIRST TIME
               JUMP            RIT             [IF SO RE-INITIALIZE TABLES
AN1:           MDXO'F          1
               JPLS            AN2             [IF ANFLG=1 SET UP FOR
               MDAR'X          ANFLG           [ACCEPTING CHARACTERS AND
               MDAR'L                          [INCREMENT ANFLG
               JUMP            $DDEF2
               ARMD            $WT1            [SET UP PIVOTS IN AMRMX FOR
               MDAR'F          $CTAB           [TELETYPE INTERFACE
```

```
                MDAE            C1              [SET P0INTER T0 NEXT
                MDAR'A          MASK1           [CTAB ENTRY T0 BE
                ARMD            $CTAB           [FILLED
1AN1:           ARX0'F
                ARMD            CCNT            [CLEAR S0ME FLAGS
                ARMD            CFLAG
2AN1:           JPSR            $ICHTY          [G0 GET A CHAR FR9M THE TTY
                                2
                MDBR            $TTYC
                MDBR'A          CMASK           [BRING IT INT0 BR, MASK AND
                BRMD            CHAR            [ST9RE IN CHAR
                ARMD'0          CFLAG
                JUMP            2AN1
AN2:            MDX0'F          1'2             [D0 N9T D0 ANYTHING UNTIL
                JPLS            AN3             [LIL INCREMENTS ANFLG AFTER
                JUMP            $DDEF2          [ALL LABELS ARE IN
AN3:            MDX0'F          2'3
                JPLS            AN4             [SET UP T9 ACCEPT NUMBERS
                MDAR'X          ANFLG
                MDAR'F          $NTAB
                MDAE            C1
                MDAR'A          MASK1
                ARMD            $NTAB
1AN3:           ARX0'F
                ARMD            CCNT
                ARMD            CFLAG
                JUMP            $DDEF2
AN4:            MDX0'F          3'4             [JUST G0 DRAW UNTIL ALL
                JPLS            AN5             [NUMBERS ARE IN
                JUMP            $DDEF2
AN5:            MDX0'F          4'5
                JPLS            AN6             [ALL N9S. IN...
                JPSR            RESP0           [G0 CALCULATE RESP0NSE
                JUMP            $DDEF2
AN6:            MDX0'F          5'6
                JPLS            .               [SH9ULD N9T GET HERE
```

```
                JUMP        $DDEF2
RIT:            MDAR'F      RAWBK       [THIS ROUTINE RE-INITIALIZES
                MDAE'N      C1          [THE BAND,BAND0,AND URD TABLES
                MDAR'A      MASK1       [TO ZERO
                ARMD        CLRFG
CLRR:           MDAR'X      CLRFG       [GET NEXT DATA ENTRY
                MDX0'F      BTAB        [CHK AGAINST BOTTOM OF DATA
                JPLS        .+2
                JUMP        TAEL        [JUMP OUT IF RIT COMPLETE
                ARX0'F
                ARMD'I      CLRFG       [CLEAR OUT DATA ENTRY
                ARMD        NOUTS
                JUMP        CLRR        [RETURN TO CLEAR NEXT ENTRY
TAFL:           MDAR'F      $DATA1
                MDAE'N      C1          [THROW AWAY ERASED LINES
                MDAR'A      MASK1
                ARMD        DT1P        [PTR TO LAST NON-ZERO DATA1 ENTRY
                ARMD        NDT1E       [PTR TO NEXT ENTRY TO BE CHKED
                ARX0'F
                ARMD        ZFLAG       [NO ZEROS FOUND FLAG
                MDAR        $TEMP       [GET BOTTOM OF DATA1 SET FM CONCT
                ARMD        BDT1        [BOTTOM POINTER
                MDAR'X      BDT1        [SET TO ENTRY BELOW DATA1
TAELR:          MDAR'X      DT1P
                MDX0        BDT1        [LOOK AT NEXT DATA1 ENTRY AND
                JPLS        .+2         [CHECK AGAINST BOTTOM PTR
                JUMP        E0LPD       [BRANCH OUT TO FIXUP UP END OF TABLE
                MDAR'I      DT1P
                JPLS        TAELR
                ARAR'H
                JPLS        TAELR       [NON-ZERO ENTRY FOUND--RETURN
                MDAR        DT1P
                ARMD        NDT1E
                ARMD'0      ZFLAG       [ZERO ENTRY FOUND--SET PTRS AND FLAG
TAZE:           MDAR'X      NDT1E       [THROW AWAY ZERO ENTRY
                MDX0        BDT1        [GET NEXT NONZERO ENTRY WITH NDT1E
```

```
                JPLS        .+2
                JUMP        E0LPD       [BRANCH 0UT IF AT B0TT0M 0F TABLE
                MDAR'I      NDT1E
                JPLS        BKSTF       [JUMP T0 SWAP ENTRIES
                ARAR'H
                JPLS        BKSTF
                JUMP        TAZE        [MUST BE ZER0 ENTRY--GET NEXT 0NE
BKSTF:          MDAR'I      NDT1E       [R0UTINE T0 STUFF A N0NZER0 ENTRY
                ARMD'I      DT1P        [INT0 A PREVI0US ZER0 ENTRY
                ARX0'F
                ARMD'I      NDT1E       [CLEARS 0UT F0UND N0NZER0 ENTRY AND
                JUMP        TAELR       [RETURNS
F0LPD:          MDAR'N      ZFLAG       [R0UTINE T0 PAD 0UT TABLE WITH E0L'S
                JPAN        ENDTA       [D0 N0T D0 IF N0 ZER0 ENTRY F0UND AB0VE
ELR:            MDAR        NDT1E
                MDX0        DT1P
                JPLS        .+2         [CHECK AGAINST LAST TABLE ENTRY
                JUMP        ENDTA
                MDAR'H      C1
                ARMD'I      DT1P        [ST0RE AN E0L INT0 THIS ZER0ED ENTRY
                MDAR'X      DT1P
                JUMP        ELR         [RETURN T0 E0L PAD NEXT ENTRY
ENDTA:          MDAR'F      $DATA1
                MDAE'N      C1
                MDAR'A      MASK1
                ARMD        RAWTP       [SET UP PTR IN DATA1 SET
                MDAE'N      C1
                ARMD        LRWTP
                MDAR'F      RAWBK
                ARMD        RBP         [SET UP PTR IN NEW 2-C0NNECT SET
                MDAR        $DATA1
                MDAR'A      C1
                JPLS        .           [A TRAP IF 1ST DATA1 ENTRY IS DRAW
RAWRT:          MDAR'X      LRWTP
                MDAR'X'I    RAWTP
                MDAR'A      C1
```

```
                JPLS        RAWRT       [CHK IT- JUMP IF A  DRAW
                MDAR'I'H    RAWTP
                MDAR'A      C1          [A M0VE--CHK E0L
                JPLS        FULD        [2-C C0MPLETE G0T0 CHK C0NNECT
                MDAR'I      LRWTP
                MDAR'A      C1
                JPLS        DRFD
                MDAR'I      RAWTP
                ARMD'I      RBP
                JUMP        RAWRT
DRFD:           MDAR'X      RBP
                MDAR'I      LRWTP
                ARMD'I      RBP         [ST0RE LAST DRAW IN 2-C SET
                MDAR'X      RBP
                MDAR'I      RAWTP
                ARMD'I      RBP         [ST0RE THE M0VE IN 2-C SET
                JUMP        RAWRT       [GET NEXT ENTRY
FULD:           MDAR'I      LRWTP       [FIX UP LAST DRAW
                MDAR'A      C1          [IF LAST DATA1 ENTRY IS DRAW
                JPLS        .+2         [ST0RE IT AS END 9F C0NNECT
                JUMP        CANAL       [FR9M THE PREVI0US M0VE
                MDAR'X      RBP         [JUMP CANAL EXECUTED IF LAST
                MDAR'I      LRWTP       [ENTRY IS A M9VE, 0THERWISE
                ARMD'I      RBP         [FALL THR0UGH...

CANAL:          MDAR'X      RBP
                MDAR'A      MASK1
                ARMD        BTRBK       [SET PTR T0 0NE BEL0W RAWBK
                MDAR'F      RAWBK       [I. E. THE B0TT0M
                MDAE'N      C1
                MDAR'A      MASK1
                ARMD        TPRBK       [SET PTR T0 T9P 0F RAWBK
                MDAR'F      BAND
                MDAR'A      MASK1
                ARMD        BPB         [PTR T0 CURRENT ENTRY IN BANAL TABLE
                MDAR'F      BAND0
```

```
                MDAR'A          MASK1
                ARMD            BPB0            [PTR TO CURRENT ENTRY IN BAND0 TABLE
                MDAR'F          URD
                MDAR'A          MASK1
                ARMD            BURD            [INIT. B0T 0F URD T0 T0P 0F URD
                MDAR'F          URI
                MDAR'A          MASK1
                ARMD            URIPT
                MDAR            LEVLL
                ARMD            LEVEL           [INIT. LEVEL T0 LEFT BETWEEN C0LS 1   2
                MDAR            TPRBK
                ARMD            RBP             [ (RE)INIT. PTR T0 CURR. RAWBK ENTRY
RTCAL:          MDAR'X          RBP
                MDX0            BTRSK           [INCR. AND CHK IF LAST RAWBK ENTRY
                JPLS            LSRCH           [JUMP IF N0T
                MDAR            LEVEL
                MDAE            LEVIN
                ARMD            LEVEL           [INCR LEVEL
                MDAE'N          LEVLR           [CHK IF LEVEL HAS BEEN SEARCHED
                ARAR'H
                JPAN            RTCAL-2         [RE-INITIAL RBP PTR AND D0 NXT LEVEL
                JUMP            ENDCA           [ALL LEVELS CHKD, EXIT CANAL
LSRCH:          MDAR'I          RBP
                ARAR'H
                MDAR'A          C1
                JPLS            RTCAL           [CHK E0L SEE IF ENTRY ALRDY MATCHED
                MDAR'I          RBP
                ARMD            FDRBK
                MDAE'N'H        LEVEL
                MDAE'N          LEVEL
                JPAN            .+2             [CHK IF RAWBK ENTRY IS AT THIS LEVEL
                JUMP            RTCAL
TBSCH:          MDAR'F          $TBL0
                MDAR'A          MASK1
                ARMD            TBPTR           [L0AD CURR TBL0 PTR
                MDAR            $TBL0
```

```
                MDAE            C1
                MDAR'A          MASK1
                ARMD            TBB0T           [L0AD B0TT0M 0F TBL0
RTBS:           ARX0'F
                ARMD            UPFLG
                ARMD            0TFLG           [CLEAR S0ME FLAGS
                MDAR'X          TBPTR
                MDX0            TBB0T
                JPLS            .+2
                JUMP            UNRES           [CHKD WH0LE TABLE  IF S0 UNRES0LVED
                MDAR'I          TBPTR
                MDAE'N          FDR8K
                ARMD            C0MDIF          [A W0RD C0NTAINING X   Y DIFF FM GATE
CHKX:           JUMP            .+3
                ARAR'N
                ARMD'0          0TFLG           [YIELDS ABS VAL 0F X AND SETS 0TFLG
                JPAN            .-2             [0TFLG IS MINUS ZER0 IF C0NNECT P0INT
                ARAR'H                          [IS T9 THE LEFT 0F GATE
                MDAR'A          MASK1
                MDAE'N          ADX2            [LEAVE IF C0NNECT PT T00 FAR FM GATE
                JPAN            .+2
                JUMP            RTBS
CHKY:           MDAR'H          C0MDIF
                JUMP            .+3
                ARAR'N
                ARMD'0          UPFLG           [SIMILAR T9 AB0VE F0R Y DIFF
                JPAN            .-2             [UPFLG IS MINUS ZER0 IF C0NNECT
                ARAR'H                          [P0INT IS AB0VE CENTER 0F GATE
                MDAR'A          MASK1
                ARMD            YDIFF
                MDAE'N          ADY2
                JPAN            .+2
                JUMP            RTBS
                MDAR'I          RBP             [CLEARS 0UT X,Y FR0M RAWBK
                MDAR'A          RXYM            [W0RD F0R FIXUP
                ARMD'I          RBP
```

```
BANAL:      MDAR'I          TBPTR           [BUILDS ANALYSIS DATA BL8CKS
            MDAR'A'L;       77000JH
            MDAR'8'I        RBP             [FIXUP RAWBK X
            ARMD'I          RBP
            MDAR'A'L;       77000JH
            ARAR'H
            ARMD'I          BPB             [STUFF BAND X
            MDAR'I          TBPTR
            MDAR'A'L;       77000
            MDAR'8'I        RBP             [FIXUP RAWBK Y
            ARMD'I          RBP
            MDAR'A'L;       77000
            ARRS            6               [THIS SECTI8N EXTRACTS 6 BIT REPN
            MDAR'8'I        BPB             [9F X VALUE 8F GATE,Y VALUE 8F GATE
            ARMD'I          BPB             [AND TYPE 8F GATE AND STUFFS IT INT8
            MDAR'I          TBPTR           [BANAL TABLE
            MDAR'A          C7
            ARMD            GTYPE
            MDAR'8'I        BPB
            ARMD'I          BPB
            MDAR            GTYPE
            MDX8'F          4               [CHECK SEE IF THIS IS A F-F
            JPLS            UCHK
            MDAR            YDIFF           [IF S9 FIND THE MIDDLE INPUT
            MDAE'N'L;       200
            JPAN            .+2
            JUMP            UCHK
            MDAR'I          BPB
            MDAR'8'L;       20000JH         [AND LABEL IT WITH A 2 IN PTR
            ARMD'I          BPB
            JUMP            STJFP
UCHK:       MDAR'N          UPFLG           [CHK JPFLAG. SET BIT TW8 IF -0
            JPAN            STJFP           [B9TT9M 8F GATE
            MDAR'I          BPB
            MDAR'8'L;       10000JH
            ARMD'I          BPB
```

```
STUFP:      MDAR            RBP         [STUFF A P0INTER T0 THE C0NNECTED
            ARMD            CPTR        [ENTRY IN RAWBK INT0 THE UPPER
            MDAR'I          RBP         [HALF W0RD
            MDAR'A          C1
            JPLS            AB0V
            MDAR'X          CPTR        [C0NNECTED W0RD BEL0W CURRENT W0RD
            JUMP            STUF
AB0V:       MDAR            CPTR        [C0NNECTED W0RD AB0VE
            MDAE'N          C1
            ARMD            CPTR
STUF:       MDAR'H          CPTR
            MDAR'0'I        BPB
            ARMD'I          BPB
            MDAR'N          0TFLG
            JPAN            N0R0T
            MDAR'H'I        BPB         [BANAL W0RD IS BUILT IN L0WER HALF
            ARMD'I          BPB0        [W0RD.  IF AN 0UTPUT, HALF-W0RD
            ARX0'F
            ARMD'I          BPB
            MDAR'X          BPB0        [R0TATE AND ST0RE IN BAND0
            JUMP            .+2
N0R0T:      MDAR'X          BPB         [R0TATI0N N0T REQUIRED (INPUT )
            JUMP            D0AGN
UNRES:      MDAR'F          URD         [UNRES0LVED ENTRY, UNABLE
            MDAE'N          C1          [T0 MAP T0 A GATE
            MDAR'A          MASK1
            ARMD            UNRPT       [ENTER IN URD TABLE
1UNR:       MDAR'X          UNRPT
            MDX0            BURD
            JPLS            .+2
            JUMP            2UNR
            MDAR'I          UNRPT
            MDX0            FDRBK       [CHECK IF AN0THER URD ENTRY
            JPLS            1UNR        [HAS SAME C00RDINATES
            JPSR            BURI        [IF S0 BUILD SIMILAR URI ENTRY
            JUMP            D0AGN       [RETURN T0 L00K AT NEXT RAWBK
```

```
2UNR:   MDAR        FDRBK       [IF THIS URD IS N9T F9UND THEN BUILD
        ARMD'I      UNRPT       [A NEW URI ENTRY
        JPSR        BURI
        MDAR'X      UNRPT
        ARMD        BURD
D9AGN:  MDAR'H      C1          [SET E9L F9R THIS C9NNECT W9RD (IT WAS
        MDAR'9'I    RBP         [CHECKED) RETURN T9 CHK NXT CNT W9RD
        ARMD'I      RBP
        JUMP        RTCAL
ENDCA:  MDAR'X      ANFLG       [ALL ANALYSIS TABLES BUILT INCREMENT
        MDAR        URIPT       [ANFLG AND G9 GET LABELS
        ARMD        BTURI
        JUMP        $DDEF2

BURI:   JUMP        .           [THIS SEGMENT BUILDS A URI ENTRY
        MDAR        UNRPT
        ARAR'H
        ARMD'I      URIPT       [ST9RE URD ADDRESS IN UPPER URI
        MDAR'I      UNRPT
        MDAR'A      C1          [TEST IF THIS URD IS AN 9UTPUT
        JPLS        .+2
        JUMP        1BUR        [IF S9 SET BIT 0 IN URI TABLE
        MDAR'X      N9UTS       [T9 S9 INDICATE
        MDAR'L;     40000JH
        MDAR'9'I    URIPT
        ARMD'I      URIPT
1BUR:   MDAR        RBP         [GET LAST TW9 DIGITS 9F RAWBK ADDR
        MDAR'A      C77
        MDAS'F      ANSW        [INDEX INT9 ANSW
        MDAR'A      MASK1
        MDAR'9'I    URIPT       [ST9RE ANSW P9INTER IN URI
        ARMD'I      URIPT
        MDAR'X      URIPT
        MDIR        BURI        [RETURN T9 CALL
```

```
RESP0:      JUMP        0           [ROUTINE TO SIMULATE CIRCUIT
            MDAR'F      BAND
            MDAE'N      C1
            MDAR'A      MASK1
            ARMD        QPTR
            MDAR'F      NBAND
            MDAR'A      MASK1
            ARMD        NBP
COPB:       MDAR'X      QPTR        [FIRST MAKE A COPY OF BAND
            MDX0        BPB         [FOR THIS ANALYSIS
            JPLS        .+2
            JUMP        NBSCH
            MDAR'I      QPTR
            ARMD'I      NBP
            MDAR'X      NBP
            JUMP        COPB
NBSCH:      MDAR        NBP
            ARMD        NBBOT
            MDAR'F      NBAND
            MDAR'A      MASK1
            ARMD        NBP
NBSR1:      MDAR'F      STACK       [SET UP STACK POINTER
            MDAE'N      C1
            MDAR'A      MASK1
            ARMD        STKPT
ZSTK:       MDAR'X      STKPT       [NOW CLEAR OUT THE STACK
            MDX0'F      STKBT
            JPLS        .+2
            JUMP        NBG0-3
            ARX0'F
            ARMD'I      STKPT
            JUMP        ZSTK
            MDAR'F      STACK
            MDAR'A      MASK1
            ARMD        STKPT
NBG0:       MDAR'I      NBP
```

```
                MDAR'A          MASK1           [SAVE GATE IDENTIFIER
                ARMD            ID
                MDAR'A          C7
                ARMD            8P              [SAVE GATE TYPE
                MDAR'I'H        NBP
                MDAR'A          C77
                MDAE'L
                MDAR            ANSW
                ARIR                            [L8AD VALUE 8F PR9PER ANSW L8C.
                ARMD'I          STKPT           [ST8RE IN STACK
                MDAR'I          NBP             [EXTRACT HEIGHT 8F THE INPUT
                MDAR'A'L;       30000JH         [ST8RE THIS IN THE STACK F8R
                MDAR'8'I        STKPT           [FLIPFL8P CALCULATI8NS
                ARMD'I          STKPT
                MDAR'X          STKPT
                MDAR            NBP
                ARMD            LNBP
NBSR2:          MDAR'X          NBP
                MDX8            NBB8T           [SEE IF WE VE L88KED AT ALL ENTRIES
                JPLS            .+2
                JUMP            GTLFF           [IF S9 G8 GET LAST F-F VALUES
                MDAR'I          NBP             [IF N8T C8NTINUE NBAND SEARCH
                JPLS            .+2
                JUMP            NBSR2
                MDAR'A          MASK1
                MDX8            ID
                JPLS            NBSR2
                MDAR'I'H        NBP
                MDAR'A          C77
                MDAE'L
                MDAR            ANSW
                ARIR                            [GET NEXT ANSW VALUE F8R
                ARMD'I          STKPT           [SAME ID ENTRY
                MDAR'I          NBP
                MDAR'A'L;       30000JH
                MDAR'8'I        STKPT
```

```
                ARMD'I          STKPT           [PUT THIS REPEATED ENTRY
                MDAR'X          STKPT           [IN THE STACK
                ARX0'F
                ARMD'I          NBP
                JUMP            NBSR2           [LOOK FOR ANOTHER REPEAT
GTLFF:          MDAR            0P              [GET LAST F-F VLAUE
                MDAE'N'L;       3               [IS THIS GATE A F-F
                JPAN            COMP            [IF NOT GO SIMLATE
                MDAR'F          $FFDT           [IF S0...
                ARMD            XTP
XXRT:           MDAR'X          XTP             [SEARCH FFDT FOR THIS F-F
                MDX0            $FFDT
                JPLS            .+2
                JUMP            COMP
                MDAR'I'H        XTP
                MDX0            ID
                MDAR'A          CM77            [MATCHES ID WITH F-F ID
                JPLS            XXRT            [TWO DIGITS AT A TIME
                MDAR            ID
                ARLS            6
                N00P
                MDX0'I          XTP
                MDAR'A          CM77
                JPLS            XXRT
                MDAR'I          XTP
                MDAR'A'L;       100001          [EXTRACT LAST F-F-VALUE AND
                ARMD            STACK-1         [UNDEF FLAG AND STORE ABOVE STACK
COMP:           MDAR'F          STACK
                MDAR'A          MASK1           [FIRST SET UP ANOTHER STACK POINTER
                ARMD            SP1
                MDAR            0P              [LOAD THIS OPERATION
                MDAE'N          C1
                MDAE'L
                JPSR'I          OPNS            [JUMP TO A ROUTINE TO DO IT
                ARIR
OUTP:           MDAR'F          BAND0           [SIMULATION DONE NOW STORE RESULT
```

```
                MDAE'N          C1
                MDAR'A          MASK1
                ARMD            0TPTR
0TSCH:          MDAR'X          0TPTR           [FIND ALL 0UTPUTS 0F THIS
                MDX0            BPB0            [GATE AND STUFF THE RESULT
                JPLS            .+2             [INT0 THE PR0PER ANSW L0C.
                JUMP            X0UT
                MDAR'I'H        0TPTR
                MDAR'A          MASK1
                MDX0            ID
                JPLS            0TSCH
                MDAR'I          0TPTR           [SAME ID F0UND
                MDAR'A          C77
                MDAS'F          ANSW
                MDAE'L
                ARMD            0
                ARMD            .+2
                MDAR            STACK
A1:             0               0               [ANSWER IS STUFFED INT0 TW0
                MDAR            .-1
                MDAE'N          C1              [ANSW L0CATI0NS
                ARMD            .+2
                MDAR            STACK
A2:             0               0
LFFCP:          MDAR            0P
                MDAE'N'L;       3               [IF THIS SIMULATED A F-F
                JPAN            0TSCH
                MDAR'I          0TPTR
                MDAR'A'L;       10000           [DETERMINE LEVEL 0F 0UTPUT
                JPLS            0TSCH
                MDAR'N          STACK
                MDAR'A          C1
                MDIR            A1              [AND SET 0R RESET APPR0PRIATELY
                MDIR            A2
                JUMP            0TSCH
XAUT:           MDAR            0P
```

```
                MDAE'N'L;       3
                JPAN            SUNP
                MDAR'I          XTP
                MDAR'A'L;       -100001         [SAVE F-F DISPLAY C00RDS
                ARMD'I          XTP
                MDAR            STACK
                MDAR'A'L;       100001
                MDAR'0'I        XTP             [0R IN NEW F-F VALUE
                ARMD'I          XTP
SUNP:           MDAR'X          LNBP            [IF ALL NBAND ENTRIES
                MDX0            NBB0T           [SIMULATED
                JPLS            .+2
                JUMP'I          RESP0           [RETURN T0 RESP0 CALL
                MDAR'I          LNBP            [ELSE
                JPLS            .+2
                JUMP            SUNP
                MDAR            LNBP
                ARMD            NBP
                JUMP            NBSR1           [G0 GET NEXT TYPE 0F GATE
SAND:           N00P                            [SIMULATES AN AND GATE
                MDAR'X          SP1
                MDX0            STKPT
                JPLS            .+2
                JUMP'I          SAND            [RETURN T0 CALL
                MDAR'I          SP1
                MDAR'A          STACK
                ARMD            STACK
                JUMP            SAND+1

S0R:            N00P                            [SIMULATES AN 0R GATE
                MDAR'X          SP1
                MDX0            STKPT
                JPLS            .+2
                JUMP'I          S0R             [RETURN T0 CALL
                MDAR'I          SP1
                MDAR'0          STACK
```

```
                ARMD            STACK
                JUMP            S0R+1

SINV:           N00P                            [SIMULATES INVERTER
                MDAR'N          STACK
                ARMD            STACK
                JUMP'I          SINV            [RETURN T0 CALL

SCSFF:          JUMP            .               [SIMULATES SET-CLEAR F-F
                MDAR            STACK
                MDAR'A          STACK+1
                MDAR'A          C1
                JPLS            SCSUD           [IF B0TH INPUTS SET..UNDEF
                MDAR            STACK
                MDAR'0          STACK+1
                MDAR'A          C1              [IF NEITHER INPUT SET
                JPLS            SC0N
                MDAR            STACK-1         [L0AD LAST F-F VALUE
                ARMD            STACK
                MDIR            SCSFF           [RETURN T0 CALL
SC0N:           MDAR'H          STACK+1         [0NE 0F THE INPUTS SET
                MDAR'A'L;       10000
                JPLS            .+2
                JJMP            SCSXT           [IF S +1 IS RESET TERMINAL
                MDAR            STACK+1         [THEN STACK ALREADY 0. K.
                ARMD            STACK           [ELSE ST0RE SET INPUT INT0
                JUMP            SCSXT           [T0P 0F STACK
SCSUD:          MDAR'H          C1
                ARMD            STACK           [SET UNDEF. FLAG
                MDIR            SCSFF           [RETURN.
SCSXT:          MDAR            STACK           [WE SET S0METHING
                MDAR'A'L;       -100000         [S0 CLEAR UNDEF FLAG
                ARMD            STACK
                MDIR            SCSFF           [RETURN...
STFF:           JUMP            .               [SIMULATE T0GGLE F-F
                MDAR            STACK
```

```
                MDAR'A          SMASK
                ARMD            STACK
                MDAR            STACK+1         [MASK OUT ALL OF LOWER PART
                MDAR'A          SMASK           [OF WORD EXCEPT F-F VALUE
                ARMD            STACK+1
                MDAR            STACK+2
                MDAR'A          SMASK
                ARMD            STACK+2
                MDAR            STACK
                MDAE            STACK+1         [ADD UP THE THREE INPUT LINES
                MDAE            STACK+2
                ARMD            TSUM
                JPLS            1STFF           [NON ZERO SUM JUMP
                MDAR            STACK-1         [NO INPUT FOUND, SO LOAD
                ARMD            STACK           [LAST F-F VALUE AND RETURN
                MDIR            STFF
1STFF:          MDXO'F          1               [IS IT A SINGLE INPUT
                JPLS            STFUD           [IF NOT THEN UNDEFINED
                MDAR'F          STACK-1
                ARMD            SP1
                MDAR'I'X        SP1             [FIND OUT WHICH INPUT HAS
                JPLS            .+2             [THE ONE ENTRY
                JUMP            .-2
                MDAR'I'H        SP1
                MDAR'A'L;       30000
                JPLS            1FND
                ARXO'F                          [ITS THE RESET LINE SO
                ARMD            STACK           [RESET AND
                MDIR            STFF            [RETURN TO CALL
1FND:           MDAR'I'H        SP1
                MDAR'A'L;       20000
                JPLS            2FND
                MDAR            C1              [ITS A SET INPUT SO
                ARMD            STACK           [SET TOP OF STACK AND
                MDIR            STFF            [RETURN
2FND:           MDAR            STACK           [ITS A TOGGLE INPUT
```

```
                MDAR'A'L;       -1
                ARMD            STACK
                MDAR'N          STACK-1         [GET LAST F-F VALUE
                MDAR'A          C1              [COMPLEMENTED
                MDAR'O          STACK           [OR STACK INTO THIS
                ARMD            STACK           [SINCE MAY BE UNDEF
                MDAR'H          STACK-1
                MDAR'A          C1
                ARAR'H
                MDAR'O          STACK           [SET UNDEF FLAG IF
                ARMD            STACK           [ITS SET LAST TIME
                MDIR            STFF            [RETURN TO CALL
STFUD:          MDAR'H          C1              [SET UNDEF FLAG FOR
                MDAR'O          STACK           [ILLEGAL COMBO
                ARMD            STACK
                MDIR            STFF            [RETURN TO CALL
SJKFF:          JUMP            .               [SIMULATE J-K F-F
                MDAR            STACK           [SIMILAR TO CLEAR-SET
                MDAR'A          STACK+1
                MDAR'A          C1
                JPLS            JTOG            [EXCEPT TWO INPUTS ARE
                MDAR            STACK           [LEGAL
                MDAR'O          STACK+1
                MDAR'A          C1
                JPLS            JK
                MDAR            STACK-1
                ARMD            STACK
                MDIR            SJKFF
JK:             MDAR'H          STACK+1         [SETS OR RESETS THE F-F
                MDAR'A'L;       10000
                JPLS            .+2
                JUMP            JKXIT
                MDAR            STACK+1
                ARMD            STACK
                JUMP            JKXIT
JTOG:           MDAR'N          STACK-1         [BOTH INPUTS PRESENT
```

```
                MDAR'A          C1              [S0 SET IN C0MPLEMENT
                ARMD            STACK           [0F LAST F-F VALUE
                MDIR            SJKFF
JKXIT:          MDAR            STACK
                MDAR'A'L;       -100000
                MDIR            SJKFF
ANFNS:          0               ANXIT           [FNS TABLE--EXIT-FNS1
                0               ANULL
                0               ANULL
                0               ANULL
                0               ANRST           [RESET--FNS-4
                REPEAT          13.
                0               ANULL
                ENDR
ANULL:          JUMP            .               [D0-N9THING R9UTINE
                MDIR            ANULL
ANRST:          JUMP            .               [RESET FNS PRESSED
                MDAR'L;         2               [SET ANFLG T0 2
                ARMD            ANFLG           [AND SET RESET FLAG
                ARMD'0          RSTFG
                MDIR            ANRST
ANXIT:          JUMP            .               [M0DE EXIT PRESSED
                ARMD'0          XITFG           [SET EXIT FLAG
                ARX0'F
                ARMD            ANFLG           [CLEAR ANFLG T0 G0
                MDIR            ANXIT           [T0 MAIN DISPLAY M0DE
[C0NSTANTS AND VARIABLES]
ANFLG:0
ZFLAG:0
CLRFG:0
C1:1
C7:7
C77:77
MASK1:77777
RXYM:0077700777
CPTR:0
```

```
RAWTP:0                         [MORE CONSTANTS AND VARIABLES
RBP:0
BTRBK:0
TPRBK:0
BPB:0
BPBO:0
TBPTR:0
TBBOT:0
XITFG:0
RSTFG:0
RAWBK:      0                           [TABLE OF CONNECTED PAIRS
            0                           [FROM DATA1
            0
            REPEAT      60.
            0
            ENDR
BAND:       0                           [TABLE OF INPUT CONNECTS
            0
            REPEAT      40.
            0
            ENDR
BANDO:      0                           [TABLE OF OUTPUT CONNECTS
            0
            REPEAT      20.
            0
            ENDR
URD:        0                           [TABLE OF UNRESOLVED DATA
            0
            REPEAT      20.
            0
            ENDR
URI:        0                           [TABLE OF UNRESOLVED INPUTS
            0                           [AND OUTPUTS
            REPEAT      20.
            0
            ENDR
```

```
ANSW:           0                               [TABLE 0F ANSWER VALUES
                0
                REPEAT          63.
                0
                ENDR
BTAB:           0                               [B0TT0M 0F DATA TABLES

NBAND:          0                               [C0PY 0F BAND TABLE F0R
                0                               [0NE TIME ANALYSIS
                REPEAT          40.
                0
                ENDR
LASTK:          0
STACK:          0                               [TABLE 0F VALUES F0R
                0                               [SIMULATI0N R0UTINES T0
                REPEAT          10.             [0PERATE 0N
                0
                ENDR

STKBT:          0

0PNS:           SAND                            [TABLE 0F SIMULATI0N R0UTINES
                S0R
                SINV
                STFF
                SCSFF
                SJKFF

[M0RE C0NSTANTS AND VARIABLES

UNRPT:0
LEVEL:0
LEVLL:65076
LEVLR:22200
LEVIN:7200
FDRBK:0
```

```
C8MDIF:0                        [M8RE C8NSTANTS AND VARIABLES
8IFLG:0
UPFLG:0
ADX2:1200
ADY2:1100
DT1P:0
NDT1E:0
BDT1:0
LAST:0
LRWTP:0
CPTR:0
NBP:0
NBR8T:0
STKPT:0
ID:0
8P:0
LNBP:0
8TPTR:0
SP1:0
BURD:0
CCNT:0
CFLAG:0
CMASK:177
CHAR:0
URIPT:0
8TURI:0
N8UTS:0
YDIFF:0
GTYPE:0
XTP:0
CM77:77000
SMASK:7777700001
TSUM:0

MD05=25000JH  [USER DEFINED INSTRUCTI8NS
```

```
MD06=26000JH
MD07=27000JH
MD10=30000JH
MD11=31000JH
AR07=27100JH

TERMINATE                    [END OF ANALR (ANALYSIS MODE)•••••••
```

```
EXPUNGE
TITLE INTR0                     [THIS PR0GRAM IMPLEMENTS INTR0DUCTI0N
ENTRY INTR0,IFCLR,IE0LR,INFNS,IEXIT,NPAGE,IRITE
ENTRY PGTAB
MACR01          P(A1,A2,A3,A4,A5)
                A1/B/K + A2/B + A5/B/K +A3/B/K +A4/B +0
                ENDM
                ZZZ=101         [ASCII C0DE DEFINITI0NS
IREPEAT ZZ, (A,B,C,D,E,F,G,H,I,J,K,L,M,N,0,Q,Q,R,S,T,U,V,W,X,Y,Z)
ZZ\.=ZZZ
ZZZ=ZZZ+1
ENDI
P.=120

INTR0:          ARX0'F
                ARMD            IXFLG           [CLEAR EXIT FLAG
                ARMD            DUNF
                MDAR'F          IFCLR           [L0AD UP F/C PIV0T
                ARMD            77755
                MD10'0'L;       61400/H         [TURN 0N SC0PE,AVG1,F/C
                JPSR            $FIN            [G0 CHECK FNS
                                INFNS
                MDAR            PAGE            [INDEX T0 CURRENT PAGE
                MDAS'F          PGTAB
                MDAR'A          MASK1
                MDAE'L
                MDAR            0
                ARIR'F
                ARMD            IARG            [ST0RE PAGE ADDR IN IARG
                JPSR            IRITE
IARG:                           0               [G0 WRITE THIS PAGE
                MDAR'N          DUNF
                JPAN            .-1             [WAIT T0 FINISH
                MDAR            IXFLG           [CHECK EXIT
                JPAN            $DDEF           [IF S0 G0 T0 MAIN DISP M0DE
                JUMP            INTR0           [ELSE G0 WRITE PAGE AGAIN
```

```
IE0LR:      N00P        .           [END 0F LIST HANDLER
            ARMD        ISAV
            ARMD'0      ITFLG
            MDAR        ISAV
            JUMP'I      IE0LR
IFCLR:      N00P        .           [FRAME CL0CK HANDLER
            FPRI
            ARMD        IFSAV
            MDAR'X      IFCNT
            MDAR'N      ITFLG       [D0NE WITH TEXT WRITE
            JPAN        IFCND       [N0, RETURN
            MDAR        IFSET
            MDx0        IFCNT       [RIGHT N0. 0F F/C TICKS
            JPLS        IFCND       [N0, RETURN
            ARMD'0      DUNF
            ARX0'F                  [YES, RESET F/C C0UNT
            ARMD        IFCNT       [AND REFRESH
IFCND:      MDAR        IFSAV
            UPRI                    [UNFREEZE INTERRUPTS AND
            JUMP'I      IFCLR       [CLEAR INTERRUPT AND RETURN
INFNS:      0           IEXIT       [FNS TABLE--M0DE EXIT FNS-1
            REPEAT      14.
            0           NPAGE       [ANY 0THER FNS, THEN
            ENDR
            0           INULL
            0           INULL
            0           NPAGE       [INCREMENT PAGE C0UNT
IEXIT:      JUMP        .           [SET EXIT FLAG
            ARMD'0      IXFLG
            MDIR        IEXIT
NPAGE:      JUMP        .           [INCREMENT PAGE C0UNT
            MDAR'X      PAGE
            MDX0        PLIM        [CHECK AGAINST MAX PAGE N0.
            JPLS        .+3
            ARX0'F                  [IF EQUAL, THEN RESET PAGE
            ARMD        PAGE        [C0UNT AND RETURN
```

```
                MDIR            NPAGE
INULL:          JUMP            .               [D8 N8THING R8UTINE
                MDIR            INULL
PGTAB:          PAGE1           [PAGE TABLE
                PAGE2
                [PAGE3
                [PAGE4
                [A LIST 8F TEXT T8 WRITE 8N THE SC8PE
PAGE1:          P(11,26,13,50);P(23,37,I.,N.);P(T.,E.,R.,A.);P(C.,T.,I.,V.)
                P(E.,11,40,0);P(13,60,L.,8.);P(G.,I.,C.,0);P(11,45,13,70)
                P(L.,A.,B.,8.);P(R.,A.,T.,8.);P(R.,Y.,11,20);P(13,70.,22,A.)
                P(40,G.,0,0)
                P(R.,A.,P.,H.);P(I.,C.,S.,40);P(P.,R.,8.,G.);P(R.,A.,M.,40)
                P(F.,8.,R.,40);P(T.,H.,E.,40);P(S.,T.,U.,D.);P(Y.,11,24,13)
                P(74.,8.,F.,40)
                P(E.,L.,E.,M.);P(E.,N.,T.,A.);P(R.,Y.,40,L.);P(8.,G.,I.,C.)
                P(40,C.,I.,R.);P(C.,U.,I.,T.);P(S.,0,0,0)
                P(11,70,13,80.);P(B.,Y.,40,R.,);P(56,40,L.,56);P(40,J.,8.,H.)
                P(N.,S.,8.,N.);P(11,76,13,84.);P(J.,U.,N.,E.);P(40,61,71,67)
                P(62,11,26,13);P(94.,22,P.,R.);P(E.,S.,S.,40);P(F.,U.,N.,C.)
                P(T.,I.,8.,N.);P(40,S.,W.,I.);P(T.,C.,H.,40);P(61,40,T.,8.)
                P(40,R.,E.,T.);P(U.,R.,N.,56);P(40,40,A.,N.);P(Y.,40,8.,T.)
                P(H.,E.,R.,40);P(S.,W.,I.,T.);P(C.,H.,40,T.);P(8.,40,C.,8.)
                P(N.,T.,I.,N.);P(J.,E.,56,0,1)
PAGE2:          P(11,26,13,50);P(T.,H.,I.,S.);P(40,P.,R.,8.);P(G.,R.,A.,M.)
                P(40,H.,A.,S.);P(4C,F.,8.,U.);P(R.,40,D.,I.);P(S.,T.,I.,N.)
                P(C.,T.,40,M.);P(8.,D.,E.,S.);P(11,36,13,60);P(D.,E.,F.,I.)
                P(N.,E.,22,55);P(55,P.,R.,8.);P(V.,I.,D.,E.);P(S.,40,0,0,1)
IRITE:          JUMP            .               [A SUBR8UTINE T8 WRITE 8UT
                MDAR'I          IRITE           [A TEXT BL8CK
                MDAE'N          C1
                ARMD            77735
                MD10'8'L;       1400JH
                ARX8'F
                ARMD            ITFLG
                MDAR'F          IE9LR
```

```
                ARMD            77736           [IRITE CONTINUED
                MD10'A'L;       -60000JH
                MDIC'A'L;       -14
                MDIC'B'L;       10
                MDAR'N          ITFLG
                JPAN            .-1
                MDIC'A'L;       -14
                MDIR'X          IRITE           [RETURN TO CALL
[CONSTANTS AND VARIABLES
IXFLG:0
DUNF:0
PAGE:0
MASK1:77777
ISAV:0
ITFLG:0
IFSAV:0
IFCNT:0
IFSET:2
PLIM:2
C1:1

MD05=25000JH  [USER DEFINED INSTRUCTIONS
MD06=26000JH
MD07=27000JH
MD10=30000JH
MD11=31000JH
AR07=27100JH


TERMINATE                       [END OF INTRO (INTRODUCTION)
```

```
EXPUNGE
TITLE LIL      [THIS PR0GRAM IMPLEMENTS MAIN DISPLAY M0DE
ENTRY LIL,DRDEF,LUPE,FCLER,E0LER,CADD, FCEND,DDEF
ENTRY DDEF2,CTAB,DR2,DRLAB,DRW0N,T1,1DRL,2DRL,TRITE
ENTRY ATELR,ARR0W,TWITE,NTAB
ENTRY DR3,ARG,ARG2,DR4,1DR4,2DR4,3DR4,DF1FG
MACR01      P(A1,A2,A3,A4,A5)
     A1^B^K + A2^B +A5^B^K + A3^B^K + A4^B + 0
ENDM
ZZ7=101                    [ASCII C0DE DEFNS...
IREPEAT ZZ, (A,B,C,D,E,F,G,H,I,J,K,L,M,N,0,Q,Q,R,S,T,U,V,W,X,Y,Z)
  ZZ\.=ZZZ
ZZZ=ZZZ+1
ENDI
P.=120


LIL:          ARX0'F
              ARMD           DF1FG          [CLEAR FLAG T0 SKIP
DDEF:         ARX0'F                        [DRAWING INITIALLY
              ARMD           M0DE           [ZER0 M0DE--MAIN DISPLAY
DDEF2:        MDAR'F         FCLFR          [L0AD F/C PIV0T
              ARMD           77755
              ARX0'F
              ARMD           DFLG           [CLEAR REFRESH FLAGS
              ARMD           FLG2
              ARMD           ADFLG          [CLEAR ALL D0NE FLAG (FCLER)
              MDAR           M0DE           [T0 SKIP C0MMANDS IN A M0DE
              JPLS           DRW
TEXTW:        N00P                          [WRITES C0MMAND DISPLAY
              MD10'0'L;      61420^H
              MDAR'F         LPHLR
              ARMD           77760          [SET UP L/P PIV0T
              ARX0'F
              ARMD           TFLG
              MD10'A'L;      17777^H        [SHUT 0FF AVG-1
```

```
                MDIC'A'L;       77763           ;SET IC BITS FOR LCG-1
                MDAR'F          TB1-1
                ARMD            77735
                MDAR'F          TEGLR
                ARMD            77736
                MDIC'0'L;       10              ;TURN ON LCG-1
                MDAR'N          TFLG
                JPAN            .-1             ;WAIT TO DRAW COMMANDS
                MDIC'A'L;       -14
                MDAR            DF1FG           ;SKIP JUMP TO DRAW
                JPAN            DRW             ;IF THIS IS FIRST TIME
                ARMD'0          FLG2
                JJMP            ADHNG
DRW:            MD10'0'L;       61400VH         ;THIS DRAWS THE CIRCUIT,
                JPSR            DRDEF           ;LABELS,VALUES,ETC.
ADHNG:          MDAR'N          ADFLG           ;WAIT FOR A REFRESH
                JPAN            .-1
JUMPR:          MDAR            MODE            ;SELECT PROPER PROGRAM
                JPLS            .+2             ;TO RETURN TO BASED ON
                JJMP            DDEF+2          ;THE VALUE OF MODE
                MDX0'F          1
                JPLS            .+2
                JJMP            $STAR           ;RETURN TO DEFINE(LOGMM)
                MDX0'F          1'2
                JPLS            .+2
                JJMP            $CONCT          ;RETURN TO CONNECT(CONCT)
                MDX0'F          2'3
                JPLS            .+2
                JJMP            $ANAL           ;RETURN TO ANALYSIS(ANALR)
                JJMP            $INTRO          ;RETURN TO INTRODUCTION
DRDEF:          JUMP            .               ;THIS SUBROUTINE DOES
                MDAR            $TBL0           ;THE DRAWING
                ARAR'A'L;       77777
                MDX0'F          $TBL0           ;SEE IF TBL0 IS EMPTY
                JPLS            .+2
                JUMP            DRFFV           ;IF S0- DRAW F-F VALUES
```

```
                MDAR'F          $TBL0           [SET PTR T0 T9P 0F TBL0
                ARMD            TEMP1
LUPE:           MDAR'X'I        TEMP1
                MDAR'A'L;       7               [GET NEXT TBL9 ENTRY
                MDAE'N'L;       1               [AND FIND GATE TYPE
                MDAS'F          CADD
                ARMD            TEMP2
                MDAR'I          TEMP1
                MDAR'A'L;       7777677770      [GET DISPLAY C00RDS 0F
                AR07'F                          [THIS GATE
                MD11            NSCAL
                MD06            INTY
                MDAR'I          TEMP2
                MDAE'L
                MD05            0               [L0AD E0V PIV9T WITH
                ARMD            77756           [PR0PER DRAW C0MMAND
                MDAR'F          E0LER           [L0AD E9L PIV9T
                ARMD            77757
                ARX0'F
                ARMD            DFLG
                MDIR            77756           [THIS INST STARTS DRAW
                MDAR'N          DFLG
                JPAN            .-1             [WAIT T0 FINISH
                MDAR            TEMP1
                MDX0            $TBL0           [IS TBL0 C0MPLETELY DRAWN
                JPLS            LUPE            [GET NEXT GATE IF N0T
DRFFV:          MDAR'F          $FFDT           [THIS R0UTINE DRAWS THE
                ARMD            TEMP1           [FLIP-FL0P VALUES EITHER
DFVRT:          MDAR'X          TEMP1           [1, 0,,0R (QUESTI0N MARK)
                MDX0            $FFDT
                JPLS            .+2
                JUMP            DRC0N           [JUMP T0 DRAW C0NNECTI0NS
                MDAR'H'I        TEMP1           [IF F-F VALUES ALL DRAWN
                MDAR'A          C1
                JPLS            .+2
                JUMP            GET01
```

```
                MDAR'F          URDFF           [SETS UP UNDEF VALUE
                ARMD            FARG            [FOR FLIP-FLOPS
                JUMP            SETXY
GFTO1:          MDAR'I          TEMP1
                MDAR'A          C7
                MDAS'F          FFOUT
                MDAR'A          MASK1
                ARMD            FARG            [SETS UP 0 OR 1 DRAW
SETXY:          MDAR'I          TEMP1           [GET COORDS FOR VALUE
                MDAR'A'L;       -100007         [DISPLAY
                MDAE'L;         125077427       [ADD A FUDGE FACTOR
                AR07'F
                ARX0'F
                ARMD            TFLG
                JPSR            TRITE           [TEXT-WRITE THIS VALUE
FARG:           0               0
                JUMP            DFVRT
DRCON:          MD10'0'L;       61400JH         [TO DRAW THE LINES
                ARX0'F                          [ENTERED IN THE CONNECT
                ARMD            DFLG            [MODE
                MD06            INTY
                MD07            $HDXY1
                MD11            HSCAL
                MDAR'F          EOLER
                ARMD            77757
                MDAR'L
                MD05            $DATA1          [LOAD THE EOV PIVOT
                ARMD            77756
                MDIR            77756           [DRAW THE CONNECTIONS
                MDAR'N          DFLG
                JPAN            .-1
                MDAR            MODE
                MDX0'L;         2
                JPLS            DRWON           [SKIP CURSOR DRAW AND
                ARX0'F                          [SELECTED VECTOR REDRAW
                ARMD            DFLG            [IF NOT IN CONNECT MODE
```

```
                MDAR            $C11            [REDRAW SELECTED LINE
                JPLS            DRCUR
                MDAR'L
                MDO5            $DATA2
                ARMD            77756
                MDIR            77756
                MDAR'N          DFLG
                JPAN            .-1
DRCUR:          MDC7            $HDXY
                ARX0'F
                ARMD            DFLG
                MDAR'L
                MDO5            $DATA           [DRAW THE CR0SSHAIRS
                ARMD            77756           [C0NNECT CURS0R
                MDIR            77756
                MDAR'N          DFLG
                JPAN            .-1
DUNDR:          ARMD'0          FLG2            [EXIT FR0M THE DRAW M0DE
                MDIR            DRDEF
DRW0N:          MDAR            $ANFLG          [THE LIL PART 0F THE
                JPLS            .+2             [F0REGR0UND/BACKGR0UND
                JUMP            DUNDR           [0PERATI0N
                MDX0'F          1
                JPLS            DR2             [ANFLG=0 0R 1, D0 N0THING
                JUMP            DUNDR
DR2:            MDX0'F          1'2             [IF ANFLG = 2 THEN SET
                JPLS            DR4             [UP THE PR0PER VARIABLES
                MDAR'N          $CFLAG          [T0 ACCEPT CHARACTERS
                JPAN            DRLAB           [JUMP 0UT IF N0  LABELS YET
                ARMD            $CFLAG          [SNEAKY CLEARING 0F FLAG
                MDAR            $CCNT
                ARMD            T1ADD           [SAVE THIS CARC0UNT TEMP
                MDAR'X          $CCNT
                MDAR            $CHAR
                MDX0'F          15              [IS THIS A C/R
                JPLS            DD1
```

```
                MDAR            $CCNT           [IF THIS IS THE FIRST
                MDX8'F          1               [CHAR D8N T ACCEPT A
                JPLS            DD2             [CARRIAGE RETURN
                ARX8'F
                ARMD            $CCNT
                JUMP            DRLAB
DD1:            MDAR            T1ADD           [SELECT THE PR8PER A/N FIELD
                MDAS'F          T1              [T8 STUFF THIS CHAR INT8
                MDAR'A          MASK1
                MDAE'L
                MDAR            0
                ARIR'F
                MDAE'L
                ARLS            0               [8RIENT THIS CHAR F8R THE
                ARMD            DR2LS           [PR8PER A/N FIELD
                MDAR            $CHAR
DR2LS:          0
                N88P
                ARMD            DISL            [ST8RE SHIFTED CHARACTER
                MDAR            $CMASK
                MDIR            DR2LS           [SHIFT MASK T8 CHAR P8SIT
                ARAR'N
                MDAR'A'I        CTAB            [MASK 8UT PREVI8US CHARS
                MDAR'8          DISL
                ARMD'I          CTAB            [REST8RE NEW CHAR IN CTAB
                MDAR            $CCNT
                MDX8'F          3               [TEST IF LAST CHAR F8R
                JPLS            DRLAB           [THIS LABEL
DD2:            MDAR'H          SIZIT           [SET SIZE BITS
                ARMD'X'I        CTAB            [AND E8L IN NEXT TABLE
                MDAR            CTAB            [ENTRY
                MDAS'N'F        CTAB+2
                MDAS'F          $URD
                ARMD            TURD
                MDAR'I          TURD            [SET THE END-8F-LIST BIT
                MDAR'8'H        C1              [IN THIS URD ENTRY (CHECKED)
```

```
                ARMD'I          TURD            [RESTORE URD
                MDAR'X          TURD
                MDX0            $BURD           [TEST END OF URD TABLE
                JPLS            .+2
                JUMP            DR3             [JUMP TO SET UP FOR NUMBS
                ARX0'F                          [IF LABEL BUILD COMPLETE
                ARMD            $CCNT           [EXIT DRAW TO GET ANOTHER
                JUMP            DRLAB           [LABEL NEXT TIME
DRLAB:          MDAR'F          $URD
                MDAE'N          C1
                MDAR'A          MASK1
                ARMD            URLP            [SET UP POINTERS IN URD
                ARX0'F
                ARMD            AROFG           [ARROW DRAW FLAG
                MDAR            $ANFLG
                MDX0'F          2
                JPLS            WIN
                JPSR            TWITE           [ANFLG=2...WRITE-INPUT LABELS
                                ILAB
                JUMP            1DRL
WIN:            MDX0'F          2'4
                JPLS            WANS            [ANFLG=4...INPUT NUMBS MSG
                JPSR            TWITE
                                INUM
                JUMP            1DRL
WANS:           MDX0'F          4'6
                JPLS            1DRL            [ANFLG=6...ANSWER MESSAGE
                JPSR            TWITE
                                ANSR
1DRL:           MDAR'X          URLP            [GET NEXT URD ENTRY
                MDX0            $BURD
                JPLS            DWAGN           [SEE IF LAST ENTRY
                JUMP            DUNDR           [EXIT DRAWING IF SO
DWAGN:          MDAR'I'H        URLP
                MDAR'A          C1              [CHECK EOL OF THIS URD
                JPLS            2DRL            [SKIP ACCEPT INFO IF SET
```

```
                MDAR            ARWFG           [SEE IF ARROW SHOULD BE DRAWN
                JPAN            2DRL
                ARMD'N          ARWFG
                MDAR'I          URLP
                MDAE'N'H        FUDGE           [ADJUST DISPLAY COORDS FOR
                ARO7                            [ARROW DRAW
                JPSR            TRITE           [DRAW ARROW AT PROPER POINT
                                ARROW
2DRL:           MDAR'I          URLP
                MDAR'A          C1
                JPLS            ST0-2           [CHECK IF THIS IS AN OUTPUT
                MDAR'I          URLP            [SET DIFFERENT DISPLAY COORDS
                MDAE'N'H        FUDGE           [IF S0
                ARO7'F
                JUMP            ST0
                MDAR'I          URLP
                ARO7'F
ST0:            ARMD            CP0S
                MDAR            URLP
                MDAS'N'F        $URD
                MDAS'F          CTAB+1
                ARMD            ARG
                JPSR            TRITE           [THIS DRAWS THE LABEL
ARG:                            0               [WHICH CORRESPONDS
                MDAR'N          URFLG           [TO THIS URD ENTRY
                JPAN            CTEND           [SKIP NUMBER DRAW IF STILL
                MDAR            ARG             [ACCEPTING LABELS
                MDAS'N'F        CTAB
                MDAS'F          NTAB
                ARMD            ARG2
                MDAR            CP0S
                MDAE'N          FUDGN
                ARO7
                JPSR            TRITE
ARG2:                           0               [THIS DRAWS THE NUMBER
CTEND:          MDAR            ARG             [FOR THIS URD ENTRY
```

```
                MDX0            CTAB            [CHECK IF AT END 0F CTAB
                JPLS            .+2
                JUMP            DUNDR           [DRAW C0MPLETE IF S0
                JUMP            1DRL            [ELSE RETURN
DR3:            ARMD'0          URFLG           [THIS R0UTINE ENTERED WHEN
                MDAR'F          $URD            [ALL THE LABELS HAVE BEEN
                MDAE'N          C1              [DEFINED.  SETS UP F0R
                MDAR'A          MASK1           [ACCEPTING NUMBERS
                ARMD            DR3P
DR3R:           MDAR'X          DR3P            [THIS SEGMENT CLEARS 0UT
                MDX0            $BURD           [THE E0L BIT SET IN THE
                JPLS            .+2             [URD ENTRY WHEN LABELED
                JUMP            DR3E
                MDAR'I          DR3P            [N0W USE E0L T0 SH0W
                MDAR'A          C1              [A VALUE HAS BEEN ASSIGNED
                JPLS            DR3R
                MDAR'I'H        DR3P
                MDAR'A'L;       -1
                ARMD'I'H        DR3P
                JUMP            DR3R
DR3E:           ARX0'F
                ARMD            UNFLG           [SET NUMBER FLAG
                MDAR'X          $ANFLG          [INCREMENT ANFLG T0 4
                MDAR            $RSTFG          [CHECK RESET (FNS-4
                JPAN            $ANCHK          [START ACCEPT NUMBS IF SET
                JUMP            DUNDR           [ELSE EXIT DRAWING
DR4:            MDX0'F          2'4
                JPLS            DR5
                MDAR'N          $CFLAG          [THIS SEGMENT SIMILAR T0
                JPAN            DRLAB           [DR2 EXCEPT WE ARE N0W
                ARMD            $CFLAG          [ACCEPTING NUMBERS
                MDAR            $CCNT
                ARMD            T1ADD
                MDAR'X          $CCNT
                MDAR            $CHAR
                MDX0'F          15
```

```
                JPLS            1DR4
                JUMP            4DR4
1DR4:           MDAR            $CHAR
                MDX8'F          60              [8NLY ACCEPT BINARY INPUTS
                JPLS            .+2
                JUMP            2DR4
                MDX8'F          60'61           [8NLY ACCEPT BINARY INPUTS
                JPLS            N8G8
2DR4:           MDAR            T1ADD
                MDAS'F          T1
                MDAR'A          MASK1
                MDAE'L
                MDAR            0
                ARIR'F                          [GET NUMERICAL CHARACTER
                MDAE'L
                ARLS            0
                ARMD            DR4LS
                MDAR            $CHAR
DR4LS:          0                               [SHIFT IT T8 PR8PER A/N FIELD
                N88P
                ARMD            INNUM
                MDAR            $CMASK
                MDIR            DR4LS
                N88P
                ARAR'N
                MDAR'A'I        NTAB            [MASK THIS DISPLAY NUMBER
                MDAR'8          INNUM
                ARMD'I          NTAB
                MDAR            TNUMB           [TNUMB IS THE ACCUMULATED
                ARLS            1               [VALUE IN BINARY F8RM
                N88P                            [SHIFT IT LEFT 8NE PLACE
                ARMD            TNUMB           [T8 ALL8W NEW INPUT
                MDAR            $CHAR           [INT8 LEAST SIG. DIGIT(LSD)
                MDAR'A          C1              [GET BINARY VALUE 8F THIS
                MDAR'8          TNUMB           [DISPLAY CHARACTER AND
                ARMD            TNUMB           [ST8RE IT IN TNUMB LSD
```

```
        MDAR            $CCNT           [CHECK IF LAST DIGIT
        MDX8'F          3
        JPLS            DRLAB           [RETURN T8 GET NEXT NUMB
        MDAR            NTAB
        MDAS'N'F        NTAB+1
        MDAS'F          $URD
        MDAR'A          MASK1           [IF LAST DIGIT
        ARMD            URDAD           [SAVE THE ADDRESS 8F THIS
        MDAR'F          $URI            [URD ENTRY
        MDAE'N          C1
        MDAR'A          MASK1           [SET UP A PTR IN URI TABLE
        ARMD            URIPT
3DR4:   MDAR'X          URIPT           [THIS SEGMENT FINDS ALL THE
        MDX8            $BTURI          [UNRES8LVED ENTRIES IN URI
        JPLS            *+2             [C8RRESP8NDING T8 THIS URD
        JUMP            4DR4            [ADDRESS
        MDAR'I          URIPT
        JPAN            3DR4
        ARAR'H
        MDAR'A          MASK1           [DID WE
        MDX8            URDAD           [FIND MATCHING URI ENTRY
        JPLS            3DR4            [N8, L88K AT NEXT URI
        MDAR'I          URIPT           [YES
        MDAR'A          MASK1
        MDAE'L
        ARMD            0
        ARMD            *+2             [GET ANSW ADDRESS T8 BE STUFFED
        MDAR            TNUMB           [BRING IN ACCUMULATED VALUE
        0                               [ST8RE VALUE IN PR8PER ANSW ADDR
        JUMP            3DR4
4DR4:   MDAR'H          SIZIT
        ARMD'X'I        NTAB            [FIX UP NEXT NTAB ENTRY AND
        MDAR            NTAB
        MDAS'N'F        NTAB+2
        MDAS'F          $URD
        ARMD            TURD
```

```
                MDAR'I          TURD
                MDAR'0'H        C1              [SET E0L (CHECKED)
                ARMD'I          TURD
                MDAR'X          TURD
                MDAE            $N0UTS          [ADD IN NUMBER 0F
                MDAR'A          MASK1           [CIRCUIT 0UTPUTS
                MDX0            $BURD
                JPLS            .+2             [CHECK END 0F URD
                MDAR'X          $ANFLG          [IF END INCREMENT ANFLG
                ARX0'F                          [T0 CALCULATE RESP0NSE 0N
                ARMD            $CCNT           [NEXT PASS
                ARMD            TNUMB           [ELSE RETURN T0 GET NEXT NUMB
                JUMP            DRLAB
N0G0:           MDAR            $CCNT           [PATCHES UP CCNT F0R N0N-
                MDAE'N          C1              [BINARY ENTRY AND
                ARMD            $CCNT
                JUMP            DRLAB           [RETURNS F0R NEXT NUMBER
DR5:            MDX0'F          4'5             [ANFLG = 5.. THE RESP0NSE
                JPLS            DR6             [HAS BEEN CALCULATED.
                MDAR'F          $URI            [THIS SEGMENT FINDS EACH
                MDAE'N          C1              [CIRCUIT 0UTPUT (BIT 0 IS
                MDAR'A          MASK1           [SET IN URI)
                ARMD            DR5PT
                ARX0'F
                ARMD            NDFLD
1DR5:           MDAR'X          DR5PT           [L00K AT NEXT URI ENTRY
                MDX0            $BTURI
                JPLS            .+3
                MDAR'X          $ANFLG          [ALL SEARCHED INCR ANFLG
                JUMP            DRLAB
                MDAR'I          DR5PT
                JPAN            .+2
                JUMP            1DR5            [N0T CIRCUIT 0UTPUT GET NEXT
                MDAR'A          MASK1           [CIRCUIT 0UTPUT F0UND
                MDAE'L
                MDAR            0
```

```
                ARIR'F                          [GET VALUE FROM ANSW
                ARMD            DNUMB           [BASED ON URI POINTER
2DR5:           MDAR'F          T1+2
                MDAE'N          NDFLD           [THIS SEGMENT CONVERTS
                MDAR'A          MASK1           [EACH BINARY DIGIT IN THE
                MDAE'L                          [ANSWER INTO DISPLAY FORM
                MDAR            0
                ARIR'F
                MDAE'L
                ARLS            0
                ARMD            DR5LS
                MDAR            DNUMB
                ARRS'I          NDFLD
                NOOP
                MDAR'A          C1              [DISPLAY A ZERO OR ONE
                MDAR'0'L;       60              [FOR THIS DIGIT
DR5LS:          0                               [THIS SEGMENT ORIENTS THE
                NOOP                            [DISPLAY CHAR FOUND ABOVE
                ARMD            INNUM           [INTO THE PROPER NTAB FIELD
                MDAR            $CMASK
                MDIR            DR5LS
                NOOP
                ARAR'N
                MDAR'A'I        NTAB
                MDAR'0          INNUM
                ARMD'I          NTAB
                MDAR'X          NDFLD
                MDX0'F          3
                JPLS            2DR5
                MDAR'X          NTAB
                JUMP            1DR5-2
DR6:            MDX0'F          5'6             [NOTHING TO DO HERE BUT
                JPLS            .               [DRAW ACCUMULATED TABLES
                JUMP            DRLAB           [TRAP IF ANFLAG GREATER
EOLER:          JUMP            0               [THAN SIX
                ARMD            ASAV            [END OF LIST HANDLER
```

```
                ARMD'0          DFLG            [E0LER JUSTS SETS A FLAG
                MDAR            ASAV
                JUMP'I          E0LER
TE0LR:          JUMP            0               [TEXT EN0 0F LIST HANDLER
                ARMD            TSAV
                MD10'A'L;       77757JH         [TURN 0FF L/P
                ARMD'0          TFLG            [SET THE TEXT FLAG WE ARE
                MDAR            TSAV            [HANGING 0N AND RETURN
                JUMP'I          TE0LR
LPHLR:          N00P                            [LIGHT PEN HANDLER
                ARMD            LSAV
                MDAR'L;         1               [DETERMINES WHICH C0MMAND
                ARMD            M0DE            [WAS BEING DRAWN WHEN
                MDAR            77735           [LIGHT PEN INTERRUPT
                MDAR'A'L;       77777           [0CCURED AND SETS M0DE
                ARAR'N'F                        [T0 APPR0PRIATE VALUE
                MDAS'F          TB2
                JPAN            .+2
                JUMP            LPEND           [EXIT--M0DE=1
                MDAR'X          M0DE
                MDAR            77735
                MDAR'A'L;       77777
                ARAR'N'F
                MDAS'F          TB3
                JPAN            .+2
                JUMP            LPEND           [EXIT--M0DE=2
                MDAR'X          M0DE
                MDAR            77735
                MDAR'A          MASK1
                ARAR'N'F
                MDAS'F          TB4
                JPAN            .+2
                JUMP            LPEND           [EXIT--M0DE=3
                MDAR'X          M0DE            [EXIT--M0DE=4
LPEND:          MDAR            LSAV            [M0DE SET..RETURN AFTER
                JUMP'I          LPHLR           [CLEARING INTERRUPT
```

```
FCLER:    JUMP          0          [FRAME CL0CK HANDLER
          FPRI
          ARMD          BSAV       [REFRESHES EVERY FSET TICKS
          MDAR'X        FCNT
          MDAR'N        FLG2       [IF ALL FIGURES DRAWN
          JPAN          FCEND
          MDAR          FSET
          MDX0          FCNT       [AND PR0PER N0. 0F TICKS
          JPLS          FCEND
          ARMD'0        ADFLG      [SET ALL D0NE FLAG (REFRESH)
          ARX0'F
          ARMD          FCNT
FCEND:    MDAR          BSAV
          UPRI                     [UNFREEZE INTERRUPTS,CLEAR
          JUMP'I        FCLER      [THIS INTERRUPT AND RETURN
TRITE:    JUMP          .          [A SUBR0UTINE F0R WRITING
          MDAR'I        TRITE      [TEXT BL0CKS USING AVG1
          MDAE'N        C1         [0UTPUT F0R X Y P0SITI0N
          ARMD          77735      [ASSUMES XY DESIRED IS IN
          MD10'0'L;     61400/H    [DESTINATI0N 7
          ARX0'F
          ARMD          DFLG
          ARMD          TFLG
          MDAR'F        E0LER
          ARMD          77757
          MDO5          M0VIT
          MDAR'N        DFLG
          JPAN          .-1
          MDAR'F        ATELR
          ARMD          77736
          MD10'A'L;     17777/H
          MDIC'A'L;     77763
          MDIC'0'L;     14         [TURN 0N IC 26 AND 27
          MDAR'N        TFLG
          JPAN          .-1
```

```
                MDIC'A'L;       -14             [RESET LCG1 AND
                MDIR'X          TRITE           [RETURN TO CALL
TWITE:          JUMP            .               [WRITES TEXT BLOCKS
                MDAR'I          TWITE           [SIMILAR TO TRITE BUT
                MDAE'N          C1              [ASSUMES XY POSITION
                ARMD            77735           [IS INCLUDED IN TEXT
                MD10'0'L;       1400JH
                ARX0'F
                ARMD            TFLG
                MDAR'F          ATELR
                ARMD            77736
                MD10'A'L;       -60000JH
                MDIC'A'L;       -14
                MDIC'0'L;       10              [IC 27 NOT TURNED ON..
                MDAR'N          TFLG            [NOT USING AVG1 OUTPUT
                JPAN            .-1             [FOR XY POSITIONING
                MDIC'A'L;       -14
                MDIR'X          TWITE
ATELR:          JUMP            .               [TEXT END OF LIST HANDLER
                ARMD            TSAV
                ARMD'0          TFLG
                MDAR            TSAV
                JUMP'I          ATELR

[CONSTANTS AND VARIABLES]

FCNT:0
FSET:2
FLG2:0
DFLG:0
ADFLG:0
TFLG:0
TEMP1:0
TEMP2:0
ASAV:0
BSAV:0
```

```
TSAV:0                          [MORE CONSTANTS AND VARIABLES...
LSAV:0
MODE:0
NSCAL:5770JH
HSCAL:37777JH
INTY:20000

C1:1
MASK1:77777
URLP:0
FUDGE:2200
FUDGY:0
FUDGN:600
UREFLG:0
TURD:0
T1ADD:          0
DR3P:0
TNUMB:0
URDAD:0
URIPT:0
UNFLG:0
CRPS:0
ARGFG:0
NDFLD:0
DNUMB:0
DR5PT:0
INNUM:0
DISL:0
C7:7
DF1FG:0
URDFF:          P(37,0,0,77,1) [UNDEFINED F-F VALUE
FFOUT:          P(37,10,0,60,1)             [ZERO(RESET)F-F VALUE
                P(37,10,0,61,1)             [ONE(SET)F-F VALUE
```

```
CADD:       $DADD                       [TABLE OF DATA SETS FOR
            $DORD                       [DRAWING THESE GATES..
            $DINVD                      [LOCATED IN LOGMM
            $FFTD
            $FFSD
            $FFKD

T1:         16.                         [BIT SHIFT TABLE FOR DISPLAY
            8.
            1.

CTAB:       1100100000                  [CHARACTER DISPLAY TABLE
            REPEAT          10.         [PRE-LOADED WITH SIZE AND
            1100100000                  [END OF LIST
            ENDR
NTAB:       1100100000                  [NUMBER DISPLAY TABLE
            REPEAT          10.         [SIMILARLY PRE-LOADED
            1100100000
            ENDR

TB1:        P(11,24,13,90.)             [TEXT BLOCKS TO DRAW
            P(23,23,37,0)               [COMMAND WORDS
            P(D.,E.,F.,I.)
            P(N.,E.,0,0)
TB2:        P(11,56,0,0)
            P(C.,O.,N.,N.)
            P(E.,C.,T.,0)
TB3:        P(11,80.,0,0)
            P(A.,N.,A.,L.)
            P(Y.,Z.,E.,0,)
TB4:        P(11,54,13,96.)
            P(I.,N.,T.,R.)
            P(O.,D.,U.,C.)
            P(T.,I.,O.,N.)
            P(0,0,0,0,1)
```

```
MOVIT:          0000100000                      [A DUMMY MOVE COMMAND
SIZIT:          11001
ARROW:          P(55,55,76,0,1)                 [ARROW TEXT DATA
ILAB:           P(11,56,13,90.)                 [VARIOUS MESSAGE TEXTS
                P(I.,N.,P.,U.)
                P(T.,11,72,L.)
                P(A.,B.,E.,L.)
                P(S.,0,0,0,1)
INUM:           P(11,56,13,90.)
                P(I.,N.,P.,U.)
                P(T.,11,72,0)
                P(V.,A.,L.,U.)
                P(E.,S.,0,0,1)
ANSR:           P(11,56,13,90.)
                P(23,A.,N.,S.)
                P(W.,E.,R.,0,1)
MD05=25000JH                    [USER DEFINED COMMANDS
MD06=26000JH
MD07=27000JH
MD10=30000JH
MD11=31000JH
AR07=27100JH
TERMINATE                       [END OF LIL (MAIN DISPLAY MODE).......
```

LIST OF REFERENCES

1. Educational Aids in Engineering, American Society for Engineering Education, March 1955.

2. Hauer, J. F. and Syms, G. H., "Digital Machine Simulation Programs in Electrical Engineering Education," Proceedings of Purdue 1971 Symposium on Applications of Computers to Electrical Engineering Education, p. 155-163, 26-28 April 1971.

3. Johnson, C. I., "Princples of Interactive Systems," IBM Systems Journal, v. 7, p. 147-173, 1968.

4. Licklider, J. C. R., "Man-Computer Symbiosis," IEE Transactions on Human Factors, HFE-1, p. 4-11, March 1960.

5. Coons, S. A., "An Outline of the Requirements for a Computer-Aided Design System," AFIPS Conference Proceedings, Spring Joint Computer Conference, p. 299-304, 1962.

6. Sutherland, I. E., "Sketchpad - A Man-Machine Graphical Communication System," AFIPS Conference Proceedings, Spring Joint Computer Conference, p. 329-346, 1962.

7. IBM Report 1620-EE-02X, 1620 Electronic Circuit Analysis Program (ECAP), 1965.

8. Schroer, B. J., "Expanded Capabilities of CSMP Graphics of the IBM 1130 Digital Computer," Simulation, v. 14, p. 205-214, May 1970.

9. Katzenelson, J., "AEDNET: A Simulator for Nonlinear Networks," Proceedings of the IEEE, v. 54, p. 1536-1522, November 1966.

10. Dertouzos, M. L., "CIRCAL: On-Line Circuit Design," Proceedings of the IEEE, v. 55, p. 637-654, May 1967.

11. Dertouzos, M. L., "An Introduction to On-Line Circuit Design," Proceedings of the IEEE, v. 55, p. 1961-1971, November 1967.

12. Skinner, B. F., The Technology of Teaching, Appleton-Century-Crofts, 1968.

13. Silvern, G. M. and L. C., "Programmed Instruction and Computer-Assisted Instruction--An Overview," Proceedings of the IEEE, v. 54, p. 1648-1656, December 1966.

14. Shannon, C. E., "A Symbolic Analysis of Relay and Switching Circuits," Transactions of the AIEE, v. 57, 1938.

15. Adage Graphics Terminal, Systems Reference Manual, Adage Inc., Boston, Mass.

16. Baskin, H. B. and Morse, S. P., "A Multilevel Modeling Structure for Interactive Graphic Design," IBM Systems Journal, v. 7, p. 218-228, 1968.

17. Williams, R., "On the Application of Graph Theory to Computer Data Structures," Advanced Computer Graphics, Economics, Techniques and Applications, Green, R. E. and Parslow, R. D., eds., p. 775-801, Plenum Press, 1971.

18. Busacker, R. G. and Saaty, T. L., Finite Graphs and Networks: An Introduction with Applications, McGraw-Hill, 1965.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Assoc Professor G. A. Rahe, Code 52 Ra Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	LT Robert Lee Johnson, Jr., USN 26 Tremont Street Penacook, New Hampshire 03301	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

Interactive Logic Laboratory

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*

Master's Thesis; June 1972

5. AUTHOR(S) *(First name, middle initial, last name)*

Robert Lee Johnson, Jr., Lieutenant, United States Navy

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1972	128	18

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

This thesis documents an investigation into the use of a computer graphics terminal to demonstrate the basic concepts of logical design. The areas of computer-assisted instruction, computer graphics, and computer-aided design are reviewed prior to the discussion of the creation of the INTERACTIVE LOGIC LABORATORY. The program is implemented on the Adage Graphics Terminal - 10 (AGT-10) of the Naval Postgraduate School Computer Laboratory.

The main emphasis of the program discussion is on the degree of interaction achieved by the program and its possible use as a learning aid for students of basic logical design courses. A bipartite graph is used to depict the network topology of the logic circuit and the program is quite successful in the simulation of simple logic circuits.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31408

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
KEY WORDS						
Interactive Computer Graphics						
Computer-aided design						
Computer-assisted instruction						
Bipartite graph						
Logic circuit design						